Approches par apprentissage pour l'estimation de mouvement multiframe en vidéo

Pierre Godet

université
PARIS-SACLAY

Approches par apprentissage pour l'estimation de mouvement multiframe en vidéo

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, sciences et technologies de l'information et de la communication (STIC)

Spécialité de doctorat: Traitement du signal et des images

Unité de recherche: université Paris-Saclay, ONERA, Traitement de l'information et systèmes, 91123, Palaiseau, France

Référent: Faculté des sciences d'Orsay

Thèse présentée et soutenue à Palaiseau, le 22 janvier 2021, par

Pierre GODET

Composition du jury:

David FILLIAT Professeur, ENSTA	Président
Pascal MONASSE Professeur, École Nationale des Ponts et Chaussées	Rapporteur & examinateur
Nicolas THOME Professeur, Conservatoire National des Arts et Métiers	Rapporteur & examinateur
Clément MALLET Directeur de recherche, Institut Géographique National	Examinateur
Patrick PEREZ Directeur de recherche, Valeo.ai	Examinateur
Guy LE BESNERAIS Directeur de recherche, ONERA	Directeur de thèse
Alexandre BOULCH Ingénieur de recherche, Valeo.ai	Co-encadrant
Aurélien PLYER Ingénieur de recherche, ONERA	Co-encadrant
Véronique SERFATY Docteure, Agence de l'Innovation de Défense, DGA	Invitée

Thèse de doctorat

NNT: 2021UPASG005

Remerciements

Je remercie l'ONERA et la DGA d'avoir co-financé ces travaux de thèse. Je remercie également les membres du jury, pour le temps qu'ils ont consacré à l'évaluation de cette thèse, et pour leurs retours très riches et intéressants.

Merci Alexandre, tu as toujours su être disponible et réactif, plein d'idées, de conseils et de solutions, et en particulier dans les moments les plus importants. Je me souviendrai de ces fins de journée où tu passais prendre des nouvelles, et de ces moments en salle de pause, à parler de jeux de société ou d'énigmes variées. Merci aussi pour ton optimisme débordant.

Merci Aurélien, tu m'as énormément guidé au début de cette aventure, puis tu as continué d'apporter des idées toujours intéressantes et originales, toujours avec bienveillance. Ta vision du monde et ta manière d'aborder les choses — ton travail, la recherche, ou même la vie en général — sont une source d'inspiration très précieuse.

Merci Guy, ta connaissance du domaine et tes conseils avisés sont en très grande partie responsables des réussites de cette thèse. Merci pour ton soutien et ton aide en particulier sur tous les aspects liés à la rédaction, mais aussi pour de nombreuses discussions autant scientifiques qu'artistiques, et toujours passionnantes. Merci également pour tes excellents cours d'IM3D (peut-être qu'il faut boire le café très sucré pour être un prof génial ?). C'est là qu'est né mon intérêt pour l'estimation de flot optique et qu'a commencé une sacrée aventure, particulièrement enrichissante à de nombreux points de vue.

Merci à tous ceux qui ont partagé mon quotidien dans le secteur "NB.03" et environs. Merci énormément Élise, Frédéric, Philippe, pour le temps que vous avez accordé à mes questions, quelles qu'elles soient. Merci infiniment aux âmes de la salle de pause [1] :
Joris, Guillaume B, Calum, Hélène, David, Maxime B, Nicolas, Maxime F, Marcela, Juliette, Rodrigo, Rodolphe, Soufiane, Javi, Simon, Benjamin B, Benjamin LT, Alexis, Guillaume VR, Guillaume H, Laurane, Gaston, Antoine, Louis, Rémy, mais aussi Clément de manière plus passagère, Pierre plus récemment, Flora enfin de retour, et notre Anthelme favori !

Merci aux sportifs, coureurs, nageurs, footeux, baby-footeux ou grimpeurs du midi. Mention toute particulière à Fabrice, Xavier, Pidel [2], pour le sport et pour bien plus.

Je remercie ma famille, pour leur soutien infaillible. Je remercie mes colocs [3], vous qui avez toujours su remplacer les passages difficiles par de supers moments de partage et d'amitié. Et enfin merci à toi Solène, qui partages ma vie depuis une durée supérieure à celle de la préparation d'une thèse, et qui me permets de garder le cap en toutes circonstances.

1. ok, certains étant plus acidus que d'autres à ce niveau.
2. qui détient également le titre du "meilleur régisseur-soutenance tombé du ciel qu'on aurait pu imaginer".
3. au sens large, et vous vous reconnaîtrez tous.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

Sommaire

1.1 Domaine de l'estimation de flot optique en vision par ordinateur

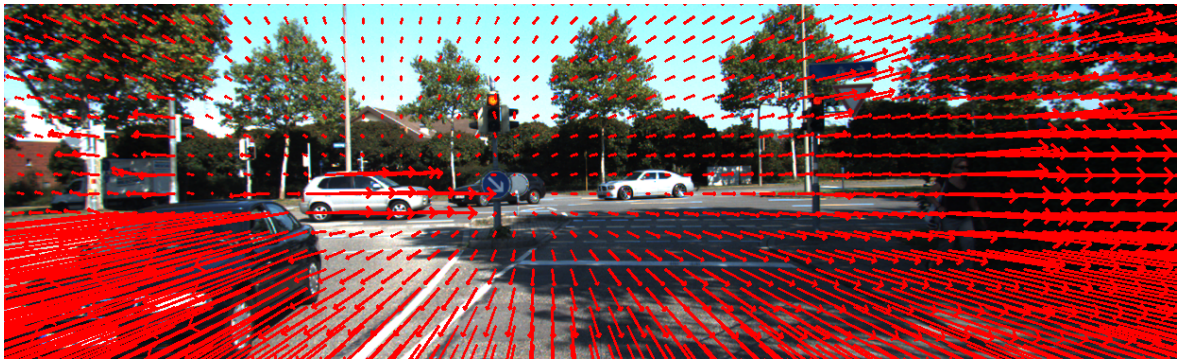

FIGURE 1.1 – Champ de mouvement apparent. Image issue du jeu de données KITTI 2015 (MENZE et GEIGER [2015]).

La vision par ordinateur vise à donner à une machine la capacité d'observer et analyser le monde à partir d'images issues d'une caméra, ou plusieurs. Elle regroupe de nombreux domaines, comme la reconnaissance et la détection de visages ou d'objets, la reconstruction 3D d'un environnement, ou encore l'estimation de mouvement.

Lorsqu'on regarde une séquence d'images acquises successivement on peut observer des mouvements des objets visibles. Ici, "objet" est à comprendre dans un sens général, on désigne par "objet" n'importe quel élément de l'image : il peut s'agir d'une voiture, ou bien du phare d'une voiture, ou encore d'une petite parcelle de la route sur laquelle la voiture roule. Le flot optique est le champ de mouvement apparent entre deux instants, dans le plan de la caméra, défini en tout pixel du champ image. La figure 1.1 montre un exemple de champ de mouvement apparent, dans le cas d'une séquence acquise par une caméra embarquée sur un véhicule qui avance. Le mouvement *apparent* est dû au mouvement relatif entre l'observateur, dans notre cas une caméra, et la scène observée. On peut distinguer, comme illustré sur la figure 1.2, deux contributions au mouvement apparent : d'une part les objets en mouvement dans le monde 3D, d'autre part l'environnement rigide, immobile dans le repère monde, dont le mouvement apparent est uniquement dû au mouvement de l'observateur.

L'estimation du flot optique présente des applications très variées. En utilisant les images acquises par une caméra fixe, par exemple dans le cas présenté en figure 1.3, on peut détecter les objets mobiles par simple seuillage sur l'amplitude du mouvement, *ie*

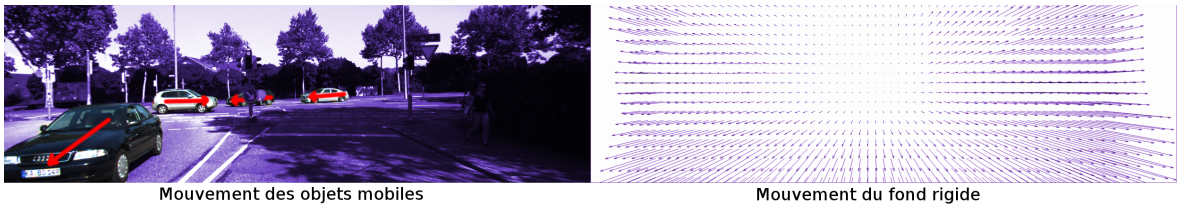

FIGURE 1.2 – Mouvement apparent : contributions des objets mobiles (flèches rouges) et de l'environnement rigide (en violet), dont le mouvement apparent (flèches violettes à droite) est dû au déplacement de la caméra.

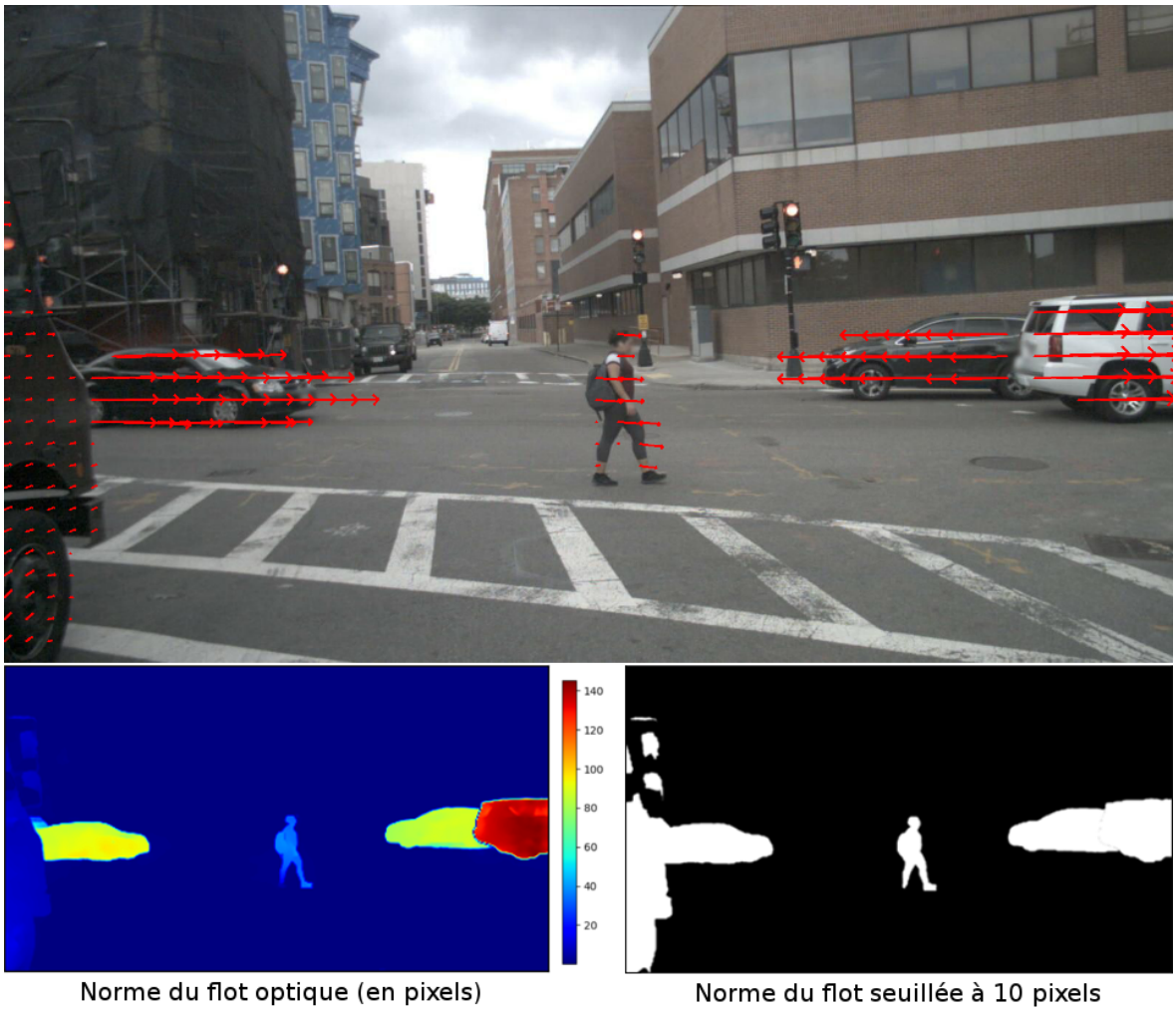

FIGURE 1.3 – Déplacement des objets mobiles entre deux instants consécutifs d'une séquence d'images acquise par une caméra fixe. Dans ce cas, on peut détecter les objets mobiles par simple seuillage sur la norme du flot optique. Image issue du jeu de données nuScenes (CAESAR et collab. [2019]).

la norme du flot optique. Grâce à l'estimation du déplacement, on peut également réaliser une prédiction de la position future de ces objets. Dans le cas d'une caméra mobile, comme dans l'exemple des figures 1.1 et 1.2, la même méthode de détection des objets mobiles nécessite au préalable de compenser le mouvement global dû au déplacement propre de la caméra. Dans le contexte de la navigation autonome, l'estimation de mouvement trouve ainsi une application pour l'évitement d'obstacles mobiles. En considérant au contraire le mouvement global de l'environnement rigide, et toujours dans le contexte de la navigation, le flot optique ouvre la voie à la "stéréo-mouvement" (*structure from motion* en anglais). À partir du mouvement apparent de l'environnement rigide, on peut déterminer le mouvement propre de la caméra et proposer une reconstruction 3D de la scène, à une transformation projective près.

Dans un contexte très différent, qui est celui de la métrologie, on peut filmer un écoulement fluide, et chercher à mesurer le champ de vitesses par vélocimétrie par images de particules ou *Particle Image Velocimetry* (PIV) à partir de la séquence d'images obtenue. Pour cela, on introduit dans le fluide des particules appelées "traceurs", et les prises de vue coïncident avec les impulsions d'un laser éclairant un plan de l'écoulement, comme

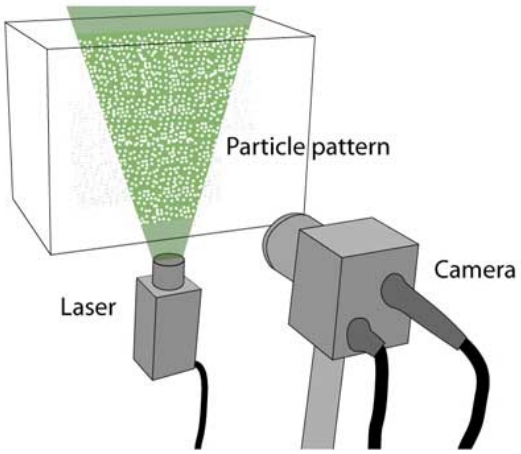

FIGURE 1.4 – Schéma illustrant un montage de PIV, pour la mesure de champ de vitesses d'un écoulement fluide. Crédit image : THIELICKE et STAMHUIS [2014].

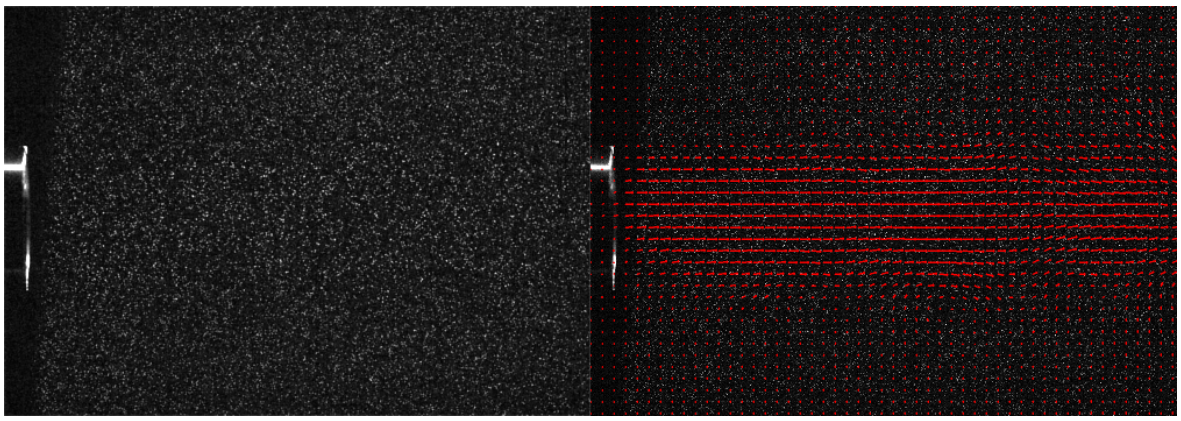

FIGURE 1.5 – Image et champ de déplacements pour une séquence de vélocimétrie par images de particules pour un jet horizontal. Exemple issus des expériences de YEGAVIAN et collab. [2016].

schématisé sur la figure 1.4. La figure 1.5 montre l'aspect très particulier des images obtenues, et le champ de déplacements estimé pour cet exemple. Les champs de vitesses estimés, et souvent leurs dérivées spatiales, sont utilisés dans les équations de la mécanique des fluides qui mettent en jeu d'autres quantités physiques d'intérêt comme la pression ou la température, et servent à la modélisation des phénomènes observés.

D'autres domaines de la vision par ordinateur utilisent également le flot optique. C'est le cas de la reconnaissance d'action, dont le but est d'analyser une séquence vidéo afin de décrire ce qu'il s'y passe, où le flot est utilisé comme une entrée supplémentaire — en plus des images. L'estimation du flot optique entre deux images permet la compensation de mouvement, c'est-à-dire le fait de rééchantillonner une des deux images pour la rendre superposable à l'autre, on parle souvent de "recalage". C'est une étape dans des méthodes d'amélioration de la qualité image, débruitage ou super-résolution, utilisant plusieurs images de la même scène.

Les premiers travaux sur l'estimation de flot optique datent des années 1980 et le domaine n'a cessé d'être actif depuis. Les méthodes proposées cherchent à mettre en correspondance des objets visibles dans les deux images : pour chaque objet de la première image on cherche la position du même objet dans la seconde image, et on en déduit le

FIGURE 1.6 – Mise en correspondance entre deux images d'une séquence vidéo. Images issues du jeu de données KITTI 2015 (MENZE et GEIGER [2015]).

déplacement. Ceci est illustré sur la figure 1.6 pour deux objets. La recherche de ces correspondances se fait classiquement en maximisant un critère de similarité sur les intensités des pixels de chaque objet, en supposant que les intensités d'un même objet restent constantes au cours du temps.

Comme le montre la figure 1.7, la recherche de correspondances entre objets d'apparences similaires est souvent ambigüe et il faut contraindre le problème par une information *a priori*, le plus souvent une notion de régularité spatiale (les pixels voisins ont des déplacements proches) et parfois aussi temporelle (les objets ont des vitesses lentement variables dans le temps). Sur ces principes de régularité, un très grand nombre de méthodes ont été proposées depuis les années 1980, à la fois pour produire des algorithmes efficaces et pour étendre les résultats à des situations de plus en plus difficiles (grands mouvements, zones homogènes, changements d'intensité, occultations, etc). Au final, aucune solution ne s'est imposée dans tous les domaines. Des algorithmes continuent à être proposés chaque année, évalués sur des jeux de données publics servant de base de référence (*benchmarks*) à la communauté scientifique.

Depuis 2015, le domaine de l'estimation du flot optique est, au même titre que le reste de la vision par ordinateur, révolutionné par les méthodes d'apprentissage par ordinateur. En quelques années, les *benchmarks* publics sont dominés par des méthodes fondées sur l'entraînement d'un réseau de neurones profond comportant plusieurs millions de paramètres sur des bases d'exemples d'apprentissage (images et flots de références)

FIGURE 1.7 – Un cas ambigu pour l'estimation de mouvement. Une estimation du flot optique entre les deux images peut présenter des ambiguïtés dans la mesure où plusieurs voitures d'aspects très semblables sont visibles, plusieurs positions donneront donc un score de similarité élevé.

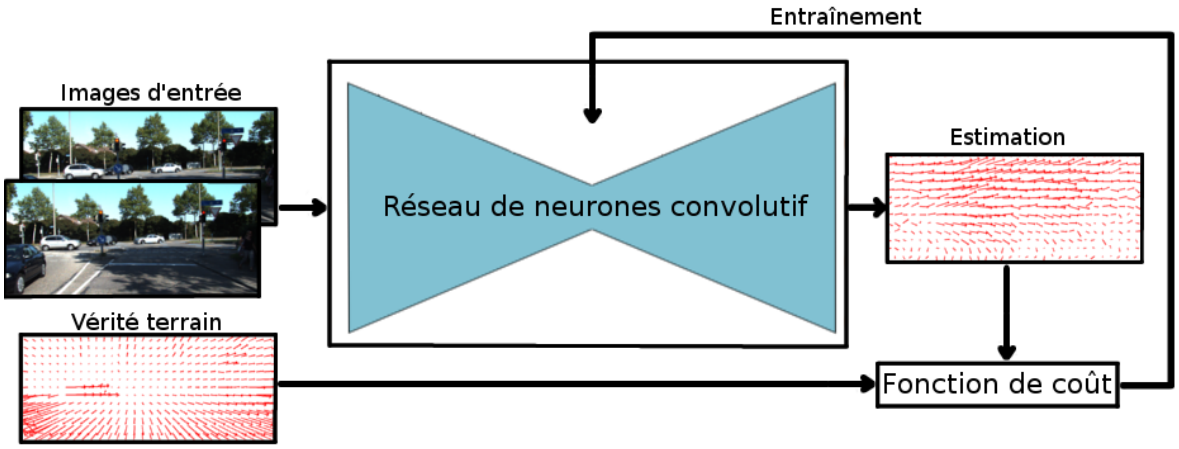

FIGURE 1.8 – Principe de l'entraînement d'un réseau de neurones pour l'estimation de flot optique.

en majeure partie synthétiques (cf. figure 1.8).

Si la majorité des travaux du domaine réalise l'estimation du flot optique à partir d'une paire d'images uniquement, d'autres proposent des méthodes dites "multiframe" exploitant des instants supplémentaires et modélisant l'évolution temporelle du mouvement. Le but de cette stratégie multiframe est de profiter de la redondance de l'information au cours du temps pour améliorer l'estimation. Dans le cas ambigu présenté sur la figure 1.7, la dynamique des objets présents, estimée sur plusieurs instants consécutifs pourrait permettre de résoudre l'ambiguité d'association. L'utilisation d'instants supplémentaires et une modélisation de l'évolution temporelle du mouvement peuvent donner accès à des informations supplémentaires sur la trajectoire des objets et ainsi améliorer la qualité de l'estimation. Néanmoins la question de l'estimation multiframe reste assez peu souvent traitée dans la littérature du domaine.

Dans la suite de ce travail, nous abordons le sujet de l'estimation du flot optique en considérant ces deux axes : l'apprentissage statistique et l'exploitation de la continuité temporelle.

1.2 Contexte de travail

Les travaux de thèse présentés dans ce manuscrit ont été réalisés à l'ONERA (Office National d'Études et de Recherches Aérospatiales), dans le département "Traitement de l'Information et Systèmes" (DTIS), au sein de l'unité IVA (Image Vision Apprentissage). L'estimation de flot optique trouve des applications dans plusieurs domaines traités par l'unité, notamment :

— la navigation autonome basée vision pour la robotique.

— la mesure de champ de vitesses par imagerie, en mécanique des fluides par exemple.

— le recalage d'images pour l'amélioration de la qualité image ou pour la fusion d'images en observation de la Terre par imagerie satellitaire ou aérienne.

Des travaux sur l'estimation de flot optique — pour la robotique, la mesure par imagerie et le recalage d'images pour l'observation de la terre — ont déjà été menés au sein de l'unité IVA, qui a notamment développé un algorithme appelé FOLKI (PLYER et collab. [2016]). Cet algorithme, basé sur le paradigme de Lucas-Kanade, donne une estimation dense du mouvement pour des coûts calculatoires très faibles. Ceci présente un intérêt particulier pour les applications concernant la robotique, dans lesquelles les algorithmes sont amenés à fonctionner sur des systèmes embarqués, aux ressources calculatoires limitées, et doivent fonctionner en temps réel. Dans ce contexte, les algorithmes d'estimation de mouvement sont donc soumis à une contrainte forte du point de vue des coûts de calculs. Plus récemment, toujours à l'ONERA, une extension multiframe de FOLKI a été proposée, nommée *Lucas-Kanade Fluid Trajectories* (LKFT) (YEGAVIAN et collab. [2016]), basée sur une modélisation temporelle polynomiale et appliquée à la mesure de champs de vitesses en mécanique des fluides. Dans ce domaine, l'accès à des moyens d'acquisition de plus en plus performants permet d'obtenir des données très résolues en temps, présentant une bonne continuité temporelle mais souvent un niveau de bruit élevé. Une méthode multiframe peut permettre d'améliorer la qualité de l'estimation, notamment en termes de robustesse au bruit.

1.3 Objectifs de la thèse

Nos travaux consistent à utiliser des techniques d'apprentissage automatique pour exploiter l'information temporelle dans un processus d'estimation du flot optique. Notre objectif est d'améliorer la qualité de l'estimation par rapport à des algorithmes ne tenant compte que de deux images consécutives — que nous appellerons "algorithmes biframe". Une analogie avec le système visuel humain motive cette idée. Si vous observez une scène pendant une fraction de seconde, vous pourrez décrire les mouvements des objets avec beaucoup moins de certitude que si vous avez pu l'observer pendant plusieurs secondes. En un temps très court, vous y verrez sans doute les mouvements principaux liés aux objets les plus étendus et occupant une position centrale dans le champ. Les zones en mouvements plus lents et les objets mobiles de petite taille en périphérie n'apparaîtront que sur une durée d'observation plus longue. Pour autant, quel gain quantitatif en termes de précision d'estimation apporte réellement l'information temporelle ? Et comment introduire la prise en compte de la continuité temporelle dans un algorithme d'estimation de flot optique ? Ces questions sont abordées tout au long de cette thèse.

Pour cela nous commençons par établir un état de l'art du domaine, dans le chapitre 2. Après avoir décrit plus précisément le domaine du flot optique et les outils associés, nous

y décrivons les méthodes classiques, dites "basées modèle", puis les méthodes plus récentes utilisant l'apprentissage profond. Enfin, nous nous intéressons aux contributions multiframe de la communauté, qu'elles soient basées modèle ou basées sur l'apprentissage profond.

Dans nos travaux nous avons exploré deux types d'approches pour réaliser l'estimation multiframe du flot optique. Le premier repose sur l'apprentissage d'une base de modèles temporels pour décrire le mouvement. Cette approche fait l'objet du chapitre 3, dans lequel nous proposons de décomposer la dépendance temporelle du mouvement sur une base apprise par analyse en composantes principales, et développons un algorithme permettant d'estimer les coefficients de cette décomposition. Ces travaux sont réalisés dans le contexte de la mesure par imagerie.

Le second consiste à utiliser l'apprentissage profond pour développer une méthode multiframe d'estimation de flot optique. Nous abordons pour cela la question des données d'entraînement annotées pour réaliser un apprentissage supervisé, et celle du développement d'architectures de réseaux de neurones. Ces questions sont traitées dans les chapitres 4 et 5. Nous considérons deux approches différentes pour tenir compte d'une information temporelle sur plusieurs instants consécutifs. La première consiste à donner à l'estimateur de mouvement un voisinage temporel de N images à traiter en une fois. Dans ce cas on peut chercher à estimer des trajectoires sur N instants, ce que nous faisons dans le chapitre 3 et au début du chapitre 4, ou bien une suite de champs de déplacements entre instants consécutifs, comme dans la seconde moitié du chapitre 4. En restant sur la problématique de l'estimation de champs de déplacements consécutifs, la seconde approche, développée au chapitre 5, exploite une séquence comme une succession de paires d'images que l'on traite au fur et à mesure avec une méthode récurrente conservant une mémoire des instants précédents.

Le chapitre 6 s'interroge sur l'utilité pratique des algorithmes fondés sur l'apprentissage au travers de comparaisons, visuelles ou quantitatives, avec des méthodes "basées modèles" sur des séquences très différentes des données d'apprentissage et relevant de divers contextes applicatifs, de la navigation autonome à la mesure de champs en mécanique. Enfin le chapitre 7 est consacré aux conclusions de ces travaux et propose des perspectives de recherche.

1.4 Contributions

Dans le chapitre 3, mené en collaboration avec Frédéric Champagnat, nous développons un algorithme, de type Lucas-Kanade, permettant de contraindre l'estimation en décomposant le mouvement sur une base arbitraire de modèles temporels. Cet algorithme généralise à une décomposition arbitraire l'algorithme LKFT qui supposait l'évolution temporelle polynomiale. En utilisant une base apprise par analyse en composantes principales à partir de la séquence étudiée, nous montrons une réduction du biais de mesure par rapport à l'estimation réalisée par LKFT.

Dans le chapitre 4 nous proposons un mécanisme simple pour tenir compte d'une séquence de $N > 2$ images dans une méthode d'estimation de flot optique basée sur un réseau de neurones, en empilant les différents instants dans le tenseur d'entrée. Afin de pouvoir réaliser un entraînement supervisé multiframe, nous générons un jeu de données d'entraînement composé de séquences annotées. Nous obtenons avec cette méthode une amélioration de la qualité de l'estimation par rapport aux méthodes biframe, dans le cas de données très résolues temporellement. Sur d'autres données, la présence de grands mouvements met en défaut la méthode.

Enfin, nous proposons STaRFlow, une nouvelle architecture neuronale récurrente pour l'estimation du flot optique multiframe, dans le chapitre 5. STaRFlow conduit à des résultats à l'état de l'art sur les *benchmarks* du domaine, tout en étant plus légère — en termes de nombre de paramètres du réseau de neurones utilisé — que les méthodes concurrentes. Nous mettons en évidence dans ce chapitre l'intérêt de l'exploitation de la continuité temporelle.

Les connaissances acquises en estimation de flot optique par apprentissage profond, au cours des travaux de recherche évoqués précédemment, ont pu servir à l'occasion d'une collaboration avec des collègues de l'unité ONERA-DTIS-IVA pour développer une méthode pour le recalage d'images issues de modalités d'acquisition différentes pour la télédétection, plus particulièrement pour le cas optique-radar. Ce travail, éloigné de l'axe principal du manuscrit, est présenté brièvement en annexe C.

1.5 Publications et communications

Les travaux réalisés ont donné lieu à plusieurs publications et communications scientifiques.

Publications acceptées ou publiées dans des conférences internationales avec comité de lecture et actes :

— **P. Godet**, A. Boulch, A. Plyer, G. Le Besnerais. "STaRFlow : A SpatioTemporal Recurrent Cell for Lightweight Multi-Frame Optical Flow Estimation". *25th International Conference on Pattern Recognition* (ICPR 2020). Acceptée.

— **P. Godet**, F. Champagnat, G. Le Besnerais, A. Plyer. "Learning fluid trajectory models for time-resolved PIV". *The 13th International Symposium on Particle Image Velocimetry (ISPIV 2019)*. Publiée.

— E. Colin Koeniguer, **P. Godet**, F. Weissgerber, B. Le Teurnier. "Multi-Sensor Coregistration by Deep Learning Applied to Sentinel-1 TerraSAR-X Images for change detection purposes". *EUSAR 2020*. "Présentation Invitée, session Deep Learning for SAR". Acceptée.

— L. Charrier, **P. Godet**, C. Rambour, F. Weissgerber, S. Erdmann, E. Colin Koeniguer. "Analysis of Dense Coregistration Methods Applied to Optical and SAR Time-Series for Ice Flow Estimations". *IEEE Radar Conference 2020*. Publiée.

Publication publiée dans une conférence nationale avec comité de lecture :

— **P. Godet**, A. Plyer, A. Boulch, G. Le Besnerais. "Estimation de flot optique multiframe par apprentissage profond". XXVIIème Colloque francophone de traitement du signal et des images (GRETSI 2019). Publiée.

Présentation orale (sans acte) :

— **P. Godet**, B. Le Teurnier, E. Colin Koeniguer, F. Janez. "Recalage d'images hétérogènes de télédétection par apprentissage profond". GDR MADICS 2019.

À venir :

— **P. Godet**, B. Le Teurnier, E. Colin Koeniguer, F.Janez, G. Le Besnerais. "Deep learning estimation of optical flow for sar/optical image registration". Article de revue en cours de rédaction.

— Nous envisageons de rédiger un article de revue sur l'estimation multiframe de flot optique par apprentissage profond.

1.6 Références

CAESAR, H., V. BANKITI, A. H. LANG, S. VORA, V. E. LIONG, Q. XU, A. KRISHNAN, Y. PAN, G. BALDAN et O. BEIJBOM. 2019, «nuScenes : A multimodal dataset for autonomous driving», *arXiv preprint arXiv :1903.11027*. 3

MENZE, M. et A. GEIGER. 2015, «Object scene flow for autonomous vehicles», dans *Conference on Computer Vision and Pattern Recognition*. 2, 5

PLYER, A., G. LE BESNERAIS et F. CHAMPAGNAT. 2016, «Massively parallel Lucas Kanade optical flow for real-time video processing applications», *Journal of Real-Time Image Processing*, vol. 11, nº 4, p. 713–730. 7

THIELICKE, W. et E. STAMHUIS. 2014, «PIVlab–towards user-friendly, affordable and accurate digital particle image velocimetry in matlab», *Journal of open research software*, vol. 2, nº 1. 4

YEGAVIAN, R., B. LECLAIRE, F. CHAMPAGNAT, C. ILLOUL et G. LOSFELD. 2016, «Lucas–kanade fluid trajectories for time-resolved PIV», *Measurement science and Technology*, vol. 27, nº 8, p. 084 004. 4, 7

Chapitre 2

État de l'Art

Sommaire

2.1 Définitions et généralités sur le flot optique

On appelle "flot optique" l'estimation dense du champ de mouvement apparent dans le plan de la caméra. Par "dense" on entend "en tout pixel de l'image", il s'agit donc de fournir un vecteur déplacement 2D par pixel. Ce vecteur est à valeurs dans $\mathbb{R}^2$, où $\mathbb{R}$ est l'ensemble des nombres réels.

2.1.1 Définition et représentation du flot optique

Le flot optique d'une image I_1, appelée "image source", vers une image I_2, appelée "image cible", est le champ de déplacements

$$\boldsymbol{u} = \begin{pmatrix} u \\ v \end{pmatrix}$$

tel que pour tout point de I_1, à la position

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2,$$

la position du point correspondant dans I_2 est donnée par

$$\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}) = \begin{pmatrix} x + u(x,y) \\ y + v(x,y) \end{pmatrix}$$

On peut représenter le champ de mouvement par des flèches dont la direction correspond à celle du mouvement et dont la longueur est proportionnelle à l'amplitude du mouvement, comme représenté sur la partie centrale de la figure 2.1. Cette représentation est très intuitive mais force à représenter une version sous-échantillonnée du champ de vecteurs pour des raisons de lisibilité. Une autre représentation couramment utilisée consiste à coder la direction du mouvement par la teinte et l'amplitude par la saturation de la couleur, voir partie basse de la figure 2.1. Cette représentation permet un affichage du mouvement de tous les pixels.

2.1.2 Conservation de l'intensité lumineuse

L'hypothèse fondamentale historique du flot optique est la conservation des intensités lumineuses le long des trajectoires. Historiquement [1], toutes les méthodes tirent leur fondement de cette hypothèse. Dans cette hypothèse de conservation des intensités, on a pour toute position $\boldsymbol{x}$ du champ image :

$$I_1(\boldsymbol{x}) = I_2(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})) \tag{2.1}$$

Pour des images discrètes de $K \times L$ pixels, on a en tout pixel de coordonnées $\boldsymbol{k} = \begin{pmatrix} k \\ l \end{pmatrix} \in S$, en notant le support image $S = \{0, ..., K-1\} \times \{0, ..., L-1\} \subset \mathbb{N}^2$:

$$\boxed{I_1(\boldsymbol{k}) = \tilde{I}_2(\boldsymbol{k} + \boldsymbol{u}(\boldsymbol{k}))} \tag{2.2}$$

Où l'on utilise l'interpolateur

$$\tilde{I}(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in S} c(k,l)\gamma(x-k)\gamma(y-l) \tag{2.3}$$

1. Avant l'apparition, en 2015, des méthodes basées sur l'apprentissage profond.

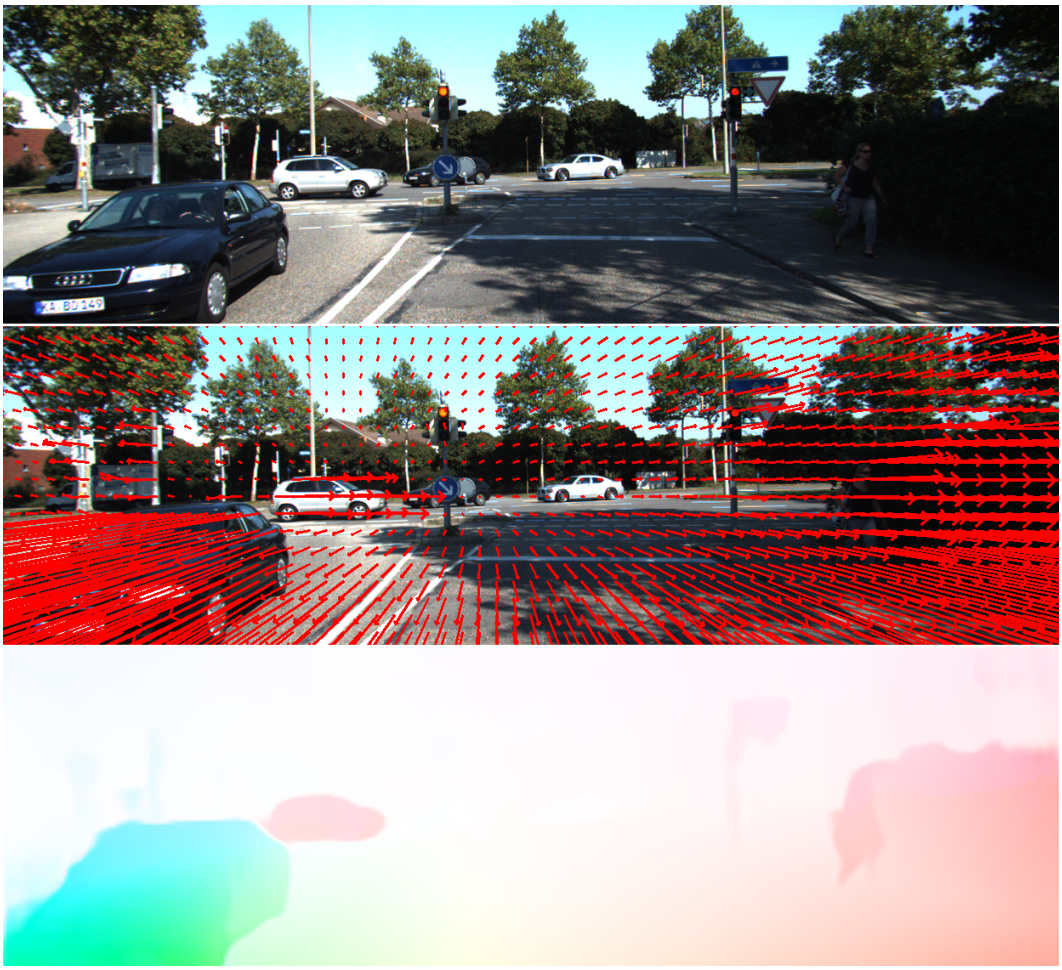

FIGURE 2.1 – Deux représentations du flot optique estimé par l'algorithme PWC-Net (SUN et collab. [2018]) sur une paire d'images de KITTI 2015 (MENZE et GEIGER [2015]). Première image de la paire (en haut), représentation sous-échantillonnée par des flèches superposées à l'image (au milieu), représentation colorée dans laquelle la teinte représente la direction et la saturation représente l'amplitude du mouvement (en bas).

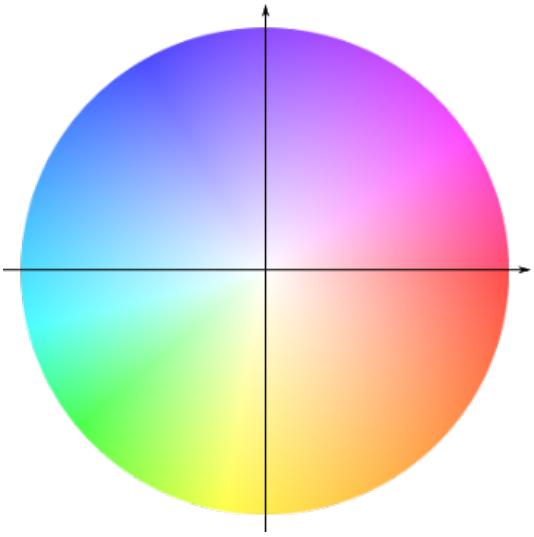

FIGURE 2.2 – Correspondance teinte-direction de la représentation colorée.

où $c(k,l)$ représente des coefficients déduits des intensités image par filtrage et γ est un noyau à support fini. Il est nécessaire d'interpoler I_2 dans la mesure où le déplacement n'est en général pas entier. Les intensités d'intérêt dans I_2 ne se situent donc pas sur la grille discrète mais "entre les pixels". La formule (2.3) d'interpolation à noyaux ci-dessus inclut des interpolateurs classiques (bilinéaires, bicubiques, splines cubiques) disponibles dans les bibliothèques de traitement d'image. Le flot optique $\boldsymbol{u}$ permet donc le recalage des images, c'est-à-dire l'interpolation de I_2 pour la rendre superposable à I_1 (dans la partie commune de leur supports).

Les exceptions à la conservation de l'intensité sont cependant fréquentes en pratique, et rendent l'estimation rigoureuse du flot optique en tout pixel impossible en général. Pour gérer ces exceptions qu'on assimile souvent à des données aberrantes (*outliers*), plusieurs méthodes sont proposées dans la littérature, comme l'utilisation de fonctions de coût robustes (BLACK et ANANDAN [1993]) ou l'utilisation de la conservation des gradients images en plus de l'intensité (BROX et collab. [2004]). Nous détaillons ci-après les différents cas d'exceptions à la conservation de l'intensité.

Exceptions liées aux changements d'intensité

Plusieurs facteurs peuvent entraîner un changement d'intensité le long d'une trajectoire. Par exemple, la luminosité globale de la scène, et donc celle d'un point d'intérêt, peut varier au cours du temps. La luminosité d'un objet mobile peut également varier si celui-ci traverse des zones de luminosités différentes, comme dans le cas du passage d'une zone ensoleillée à une zone d'ombre, représenté sur la figure 2.3.

Certains objets peuvent donner lieu à des réflexions spéculaires, qui sont particulièrement gênantes pour une méthode cherchant à mettre en correspondance les intensités. De telles réflexions peuvent changer radicalement l'aspect d'un même objet entre deux instants, en faisant apparaître un reflet très brillant, ou encore une réflexion d'une autre partie de la scène. Dans le premier cas on a des intensités sans rapport avec celle observées dans l'image de référence, dans le second on peut même faire apparaître des mouvements "fantômes" correspondant à des associations entre objets réfléchis, voire entre reflets et objets réels.

Exceptions liées à la visibilité des objets

L'hypothèse de conservation des intensités suppose que les mêmes points du monde soient visibles aux deux instants considérés. Des objets qui entrent ou sortent du champ de vue de la caméra, ou encore des objets qui passent les uns derrière les autres, entraînent des apparitions et disparitions de points entre deux instants, comme illustré en figure 2.4. Le cas d'un point visible dans l'image source mais occulté (ou sorti du champ) dans l'image cible est particulièrement gênant puisqu'il donne lieu à un pixel pour lequel le flot optique n'est pas défini par une équation de conservation (2.2). C'est le cas de la jambe du personnage sur l'exemple de la figure 2.4.

Dans la suite, on parlera d'occultation, ou zone occultée, dans le cas d'un pixel visible dans l'image source mais occulté dans l'image cible, qu'il s'agisse réellement d'une occultation entre objets, ou bien d'une sortie de champ.

FIGURE 2.3 – Exemple d'un changement d'intensité le long d'une trajectoire. Ici la voiture passe d'une zone ensoleillée à une zone ombragée. Image issue du jeu de données KITTI 2015 (MENZE et GEIGER [2015]).

FIGURE 2.4 – Une paire d'images et la carte d'occultations associée, issues du jeu de données MPI Sintel (BUTLER et collab. [2012]). Les zones en blanc sur la carte d'occultations correspondent aux points de l'image source qui ne sont pas visibles dans l'image cible. C'est le cas des points du fond occultés par des objets mobiles, mais aussi des points qui sortent du champ, ou encore d'un objet mobile qui passe derrière un autre, comme la jambe du personnage.

2.1.3 Hypothèse des petits mouvements et stratégie multi-échelle

L'estimation du flot optique est classiquement [2] effectuée en cherchant le champ de déplacements $\boldsymbol{u}$ qui minimise une fonction d'erreur de recalage :

$$\int_{\boldsymbol{x}\in\Omega} \phi\left(\|I_1(\boldsymbol{x}) - I_2(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x}))\|\right) d\boldsymbol{x} \tag{2.4}$$

où ϕ est une fonction de pondération à définir (elle est classiquement quadratique), et Ω représente l'ensemble des sites de l'image. Il est courant de linéariser cette expression par un développement limité au premier ordre de $I_2(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x}))$ autour de $\boldsymbol{x}$, ce qui suppose un déplacement $\boldsymbol{u}$ d'amplitude faible (de l'ordre du pixel). En pratique, on peut décomposer le champ de déplacements comme $\boldsymbol{u} = \boldsymbol{u}^{(0)} + \boldsymbol{\delta u}$ où $\boldsymbol{u}^{(0)}$ est une initialisation obtenue par ailleurs et $\boldsymbol{\delta u}$ est un incrément supposé petit. On réalise alors le développement limité autour de $\boldsymbol{x}+\boldsymbol{u}^{(0)}(\boldsymbol{x})$ et on cherche le $\boldsymbol{\delta u}$ qui minimise :

$$\int_{\boldsymbol{x}\in\Omega} \phi\left(\left\|I_1(\boldsymbol{x}) - I_2(\boldsymbol{x}+\boldsymbol{u}^{(0)}(\boldsymbol{x})) - \nabla I_2(\boldsymbol{x}+\boldsymbol{u}^{(0)}(\boldsymbol{x}))\boldsymbol{\delta u}(\boldsymbol{x})\right\|\right) d\boldsymbol{x} \tag{2.5}$$

Cette hypothèse de petits mouvements est très restrictive, les mouvements étant souvent plus grands que la taille du pixel en pratique. Il est alors courant de mettre en œuvre une stratégie multi-échelle, afin de pouvoir considérer des mouvements de plus grande amplitude. Pour cela, comme illustré sur la figure 2.5, on constitue une pyramide multi-échelle composée de versions des images sous-échantillonnées de plus en plus grossièrement — il faut au préalable effectuer un filtrage passe-bas des fréquences spatiales pour éviter le repliement de spectre. L'estimation du flot optique est d'abord réalisée sur les versions les plus sous-échantillonnées des images, comportant donc peu de pixels. À cette résolution grossière, les déplacements qui sont grands à pleine résolution peuvent être considérés comme petits. Le résultat obtenu est utilisé pour initialiser l'estimation au niveau suivant, pour lequel la résolution est un peu plus fine, et ainsi de suite. Les grands mouvements sont donc estimés aux échelles grossières, tandis que les petits mouvements sont estimés aux échelles plus fines. On appelle "coarse-to-fine" cette stratégie multi-échelle qui consiste à commencer l'estimation par la résolution la plus grossière avant d'affiner itérativement la résolution.

Une limite importante de cette stratégie multi-échelle est celle d'un petit objet qui se déplace rapidement. L'objet étant petit, il sera invisible aux résolutions grossières, et son mouvement sera trop grand pour être estimé aux résolutions fines.

2.1.4 Un problème inverse

En toute généralité, la minimisation de l'erreur de recalage (2.4) en chaque pixel ne présente pas de solution unique. Dans une zone homogène, ou plus généralement dans une zone présentant un gradient spatial nul dans au moins une direction, plusieurs valeurs du déplacement minimiseront (2.5). C'est aussi le cas si la scène observée est composée de plusieurs objets très semblables, entraînant plusieurs minima locaux pour (2.5). Pour résoudre cette indétermination on utilise une régularisation spatiale qui peut être locale (méthodes par corrélation de fenêtre, LUCAS et collab. [1981]) ou globale (approche variationnelle, HORN et SCHUNCK [1981]). Ces deux types d'approches, constituant l'essentiel des méthodes classiques (ou basées modèle) d'estimation de flot optique, sont détaillées dans les sections 2.2.1 et 2.2.2.

2. Nous décrivons ici les pratiques historiques utilisées dans les approches basées modèle. Nous introduirons plus tard les méthodes basées sur l'apprentissage profond, dont le principe est différent.

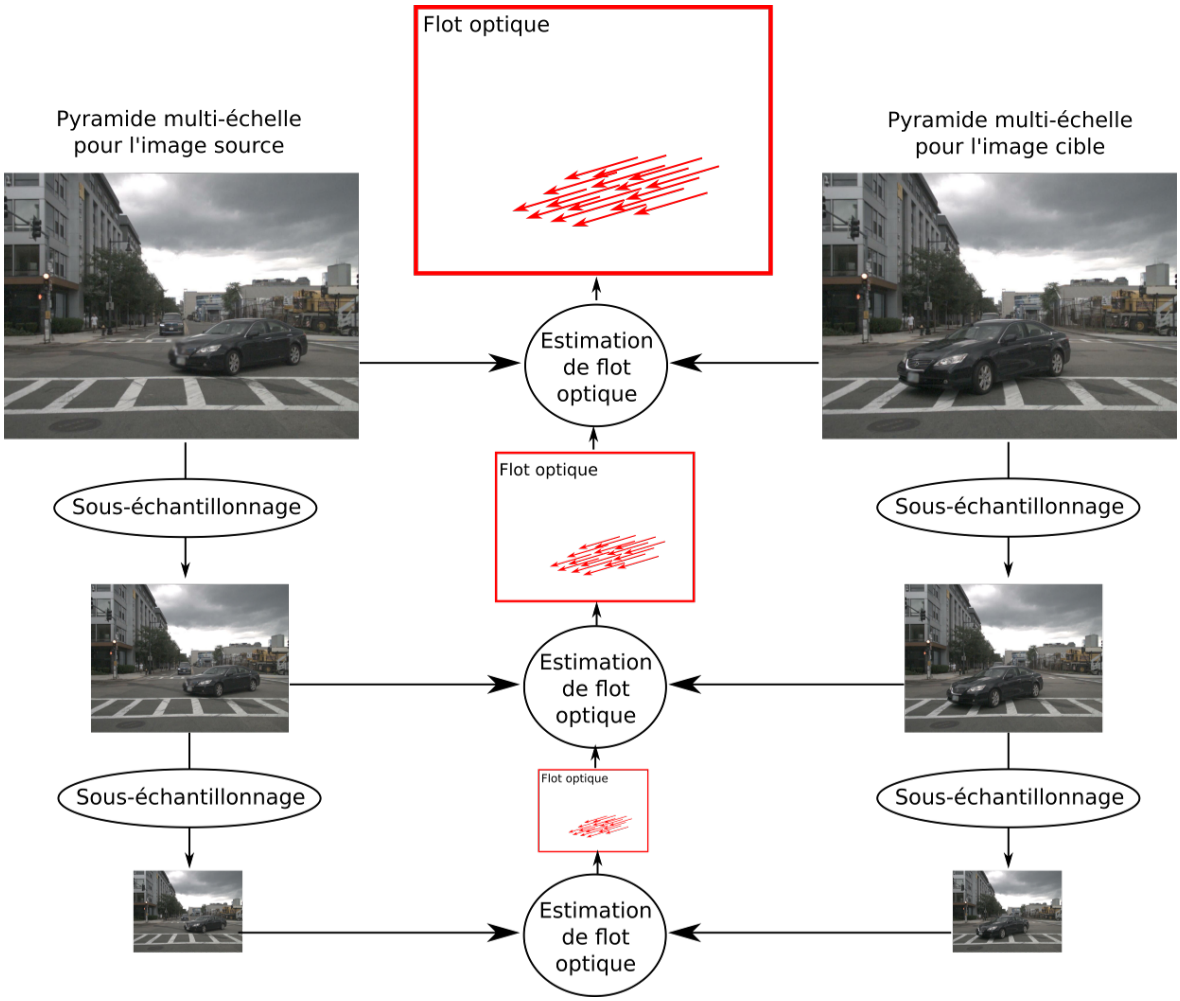

FIGURE 2.5 – Schéma de principe de l'estimation multi-échelle.

La régularisation spatiale traduit une information de lissage, c'est à dire de variation lente du champ de mouvement d'un pixel à son voisin. Elle est toujours associée à un paramètre (taille de fenêtre dans les approches locales, paramètre de régularisation dans les approches globales) qui permet de régler le compromis entre précision du recalage et douceur de la solution. Classiquement, la régularisation permet de contrôler l'influence du bruit présent dans les données sur la solution. Dans le cas du flot optique, elle permet surtout de fournir une solution au "problème d'ouverture" qui intervient dans les régions (zones homogènes, contours droits) où le champ de mouvement est (partiellement) inobservable. Le réglage est lié à ces situations et conduit à un compromis classique biais/variance. Par exemple, pour une approche locale, augmenter la taille de la fenêtre de corrélation permet de diminuer la variance de l'estimation dans les zones homogènes, mais conduit à un biais vers une solution lisse et la perte des détails fins du champ de mouvement. En pratique, les images sont souvent formées d'une juxtaposition de zones d'apparences très différentes (zones homogènes, contours, textures) et le choix d'un paramètre unique pour estimer le mouvement sur l'ensemble du champ conduit à des compromis plus ou moins satisfaisants suivant les zones.

2.1.5 Évaluation des méthodes de flot optique

Pour permettre l'évaluation et la comparaison des méthodes d'estimation de flot optique, des sites de *benchmarking* mettent à disposition des séquences d'images annotées d'une vérité terrain du flot optique, et proposent des métriques pour évaluer et classer les différentes méthodes. On peut notamment citer les jeux de données Middlebury (BAKER et collab. [2007]), KITTI 2012 (GEIGER et collab. [2012]) et 2015 (MENZE et GEIGER [2015]), et MPI Sintel (BUTLER et collab. [2012]).

L'obtention d'une vérité terrain pour le flot optique n'est pas aisée. Il n'est pas pensable d'annoter manuellement le mouvement de chaque pixel de tous les instants de plusieurs séquences. De nombreuses données utilisées pour les *benchmarks* sont par conséquent des images de synthèse (qui se veulent les plus réalistes possible), par exemple le *dataset* MPI Sintel et une partie des séquences de Middlebury. Dans ce cas, il est possible de synthétiser également la vérité terrain (VT) du flot. Il est aussi possible d'obtenir une VT en recourant à un système de mesure actif. Dans certaines séquences de mouvement d'objets non rigides de Middlebury, les objets ont été texturés à l'aide d'une peinture fluorescente invisible pour une caméra standard mais "visible" dans l'ultraviolet (UV). La scène a donc été également filmée par une caméra UV et un *tracking* a permis de construire une VT. Le *dataset* KITTI propose des images de scènes routières réelles prises par des capteurs embarqués sur un véhicule. En plus des données des caméras, on dispose de celles d'une centrale inertielle et d'un LIDAR. Une VT du flot (non dense) est obtenue à partir de ces données complémentaires.

KITTI 2012 et KITTI 2015

KITTI est un jeu de données de navigation routière. La version de 2012 ne comporte pas d'objets mobiles, le mouvement apparent n'est dû qu'au mouvement propre de la caméra. La version de 2015 comporte des objets mobiles. Chacune de ces versions comporte environ 200 séquences d'entraînement et 200 autres pour l'évaluation. Pour chaque séquence, acquise à une cadence de 10 Hz et comportant 20 images, seule la paire d'images centrale est annotée. Ces séquences présentent des cas de grands déplacements, d'occultations entre objets, de réflexions spéculaires (en particulier sur les vitres des voitures) et

de changements d'intensité le long des trajectoires (par exemple lors du passage d'un objet d'une zone ensoleillée à une zone ombragée). La figure 2.1 montre un exemple issu de KITTI 2015.

MPI Sintel

MPI Sintel est un jeu de données synthétiques obtenues à partir d'un film d'animation réalisé à l'aide du logiciel Blender. Certaines séquences du film ont été sélectionnées et regénérées avec la vérité terrain dense du flot optique et des cartes d'occultations. Les séquences proposées sont assez longues, la plupart étant composées de 50 images (quelques unes sont plus courtes), et annotées pour chaque paire d'instants consécutifs. L'ensemble d'entraînement rassemble 23 séquences, soit 1041 paires d'images annotées, l'ensemble d'évaluation comporte 12 séquences, soit 564 paires d'images. Les séquences sont de difficulté variée, présentant pour certaines des petits, des grands, voire de très grands mouvements, et plus ou moins d'occultations entre objets. Ce jeu de données, bien que simulé, se veut très réaliste, grâce à de nombreux effets de rendu jouant sur l'illumination et la réflectivité des surfaces. Deux versions, avec des options de rendu différentes, sont utilisées pour la comparaison des méthodes : Sintel Clean et Sintel Final. Sur la version "Final", des effets supplémentaires ont été appliqués : flou de bougé, flou de défocalisation et diffusion atmosphérique (effet brouillard). La figure 2.6 montre un exemple issu de Sintel.

Les métriques proposées

Afin d'évaluer et comparer différentes estimations du flot, on peut calculer diverses mesures d'erreur par rapport à la VT :

— L'erreur angulaire : `arccos` du produit scalaire entre le flot estimé et le flot VT, après normalisation des flots, en général moyenné sur tous les pixels. Cette métrique était proposée dans les comparaisons sur Middlebury, mais est peu utilisée dans la littérature actuelle, elle présente en effet l'inconvénient d'être insensible à une erreur sur la norme du flot optique.

— L'erreur *endpoint* (EPE) : norme euclidienne de l'écart à la VT, en général moyennée sur tous les pixels.

$$\mathrm{EPE}(\boldsymbol{k}) = \sqrt{(u(\boldsymbol{k}) - u_{VT}(\boldsymbol{k}))^2 + (v(\boldsymbol{k}) - v_{VT}(\boldsymbol{k}))^2} \tag{2.6}$$

$$\mathrm{EPE}_{\text{moyenne}} = \frac{1}{K \times L} \sum_{\boldsymbol{k}} \sqrt{(u(\boldsymbol{k}) - u_{VT}(\boldsymbol{k}))^2 + (v(\boldsymbol{k}) - v_{VT}(\boldsymbol{k}))^2} \tag{2.7}$$

Cette mesure est couramment utilisée dans la littérature, elle tient compte des erreurs sur la direction et sur l'amplitude. Cependant pour une même erreur relative, cette mesure est plus sensible aux erreurs sur les grands mouvements que sur les petits.

— Fl (critère de KITTI) : pourcentage des pixels pour lesquels la norme de l'écart à la VT (EPE) est supérieure à 3 pixels. Cette mesure est peu sensible à une erreur, même élevée, si celle-ci est très localisée.

Les *benchmarks* Sintel et KITTI proposent des calculs d'erreurs restreint à certaines catégories de mouvement, distinguant notamment les cas d'occultations et les cas où l'information est bien visible dans les deux images d'intérêt. Dans Sintel, l'erreur est également calculée pour plusieurs plages d'amplitude de mouvement, permettant une évaluation

FIGURE 2.6 – Un exemple issu de MPI Sintel, séquence "Alley_2". Image source (en haut) et vérité terrain représentée par des flèches (au milieu) ou dans la représentation colorée décrite en 2.1.1 (en bas).

restreinte aux petits ou aux grands mouvements, ces critères seront utilisés dans les évaluations du chapitre 4 et seront détaillées à ce moment là, en section 4.3.4.

Plusieurs méthodes d'estimation de flot optique abordent la question de la détection des zones d'occultations, qui est souvent nécessaire à une estimation correcte du flot optique. Il s'agit d'un problème de classification binaire, en chaque pixel, qui donne lieu aux métriques suivantes : précision, rappel et score F_1. Dans ce cas, la précision correspond à la proportion des pixels effectivement occultés parmi l'ensemble des pixels classifiés comme occultés, le rappel correspond à la proportion d'occultations détectées par rapport au nombre total de pixels occultés. Le score F_1 est la moyenne harmonique de la précision et du rappel et prend des valeurs entre 0 et 1, une valeur de 1 correspondant à une détection parfaite, nous l'exprimerons comme un pourcentage.

$$F_1 = 2 \times \frac{\text{précision} \times \text{rappel}}{\text{précision} + \text{rappel}} \tag{2.8}$$

$$\text{précision} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux positifs}} \tag{2.9}$$

$$\text{rappel} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux négatifs}} \tag{2.10}$$

Nous utiliserons le score F_1 pour caractériser la détection d'occultations dans le chapitre 5.
Note : la notation du critère de KITTI "Fl" est à distinguer de celle du score F_1. "Fl" correspond aux deux premières lettres de "Flow".

2.1.6 Tendance récente : le flot optique par apprentissage profond

Après de très nombreux développements autour des méthodes basées modèle, fondée sur la conservation de l'intensité, pendant près de 4 décennies, la tendance actuelle consiste à réaliser la tâche de l'estimation du flot optique par apprentissage profond en utilisant des réseaux de neurones convolutifs. Ces réseaux de neurones sont, pour la majorité des travaux sur le sujet, entraînés de manière supervisée : les paramètres des réseaux sont optimisés de sorte à minimiser l'écart entre leur sortie et la vérité terrain du flot optique. On utilise pour cela des données simulées, pour lesquelles cette vérité terrain est accessible. Dans ce cas, l'estimation n'est plus explicitement basée sur l'hypothèse de conservation des intensités lumineuses. Certains travaux proposent un apprentissage non supervisé, permettant de réaliser l'entraînement sur des données non annotées, et donc sur des données réelles. Dans ce cas, l'hypothèse de conservation des intensités redevient essentielle car elle est utilisée pour pénaliser le résultat du réseau et optimiser ses paramètres.

Les premiers travaux sur l'estimation de flot optique par apprentissage profond supervisé datent de 2015 (DOSOVITSKIY et collab. [2015]), et ces méthodes sont actuellement parmi les plus performantes sur les *benchmarks* du domaine et présentent des temps de calculs par paire d'images inférieurs à la seconde (ILG et collab. [2017]; SUN et collab. [2018]), tandis que les méthodes variationnelles les mieux placées sur ces mêmes *benchmarks* nécessitent plusieurs secondes, parfois plus d'une dizaine de secondes, de calcul par paire d'images. Les méthodes utilisant l'apprentissage profond sont présentées en section 2.3.

2.2 Estimation de flot optique par méthodes basées modèle

Ces méthodes sont basées sur l'hypothèse de conservation de l'intensité et, souvent, sur l'hypothèse des petits mouvements. Comme nous l'avons déjà mentionné, il existe des méthodes locales et des méthodes globales, introduisant des stratégies différentes de régularisation.

2.2.1 Les méthodes locales

Dans l'approche locale on suppose le flot localement constant et on cherche à estimer le mouvement d'un pixel en supposant que tous les pixels de son voisinage, dans une fenêtre donnée, suivent ce même mouvement. Pour un voisinage comportant n pixels on a donc n équations (l'équation (2.2) pour chaque pixel du voisinage) faisant intervenir les 2 composantes du même déplacement à estimer. Ceci constitue une régularisation locale et lève l'indétermination à l'échelle de la fenêtre (une zone homogène de taille beaucoup plus grande que celle de la fenêtre continuera à générer des problèmes d'association ambigüe).

La taille de la fenêtre est le paramètre clef dans cette approche. Augmenter cette taille correspond à effectuer une régularisation spatiale plus importante. On doit donc considérer une fenêtre suffisamment grande pour être robuste face au bruit et au problème d'ouverture (donc d'autant plus grand que l'image est peu texturée), mais suffisamment petite pour que le mouvement y soit homogène ou presque. Une fenêtre trop grande impliquera la perte des structures spatiales fines, tandis qu'une fenêtre trop petite risque de donner un résultat bruité.

L'algorithme de Lucas-Kanade et sa version multi-échelle PyramLK

L'approche locale a été décrite en 1981 dans LUCAS et collab. [1981] pour implémenter une méthode de recalage rapide et capable de gérer des mouvements plus complexes qu'une translation globale, comme la rotation ou le changement d'échelle. Cette méthode présente un faible temps de calcul grâce à une hypothèse de petits déplacements (en supposant par exemple un recalage approximatif préalable) et à l'utilisation du gradient spatial pour guider l'estimation et réduire la zone de recherche.

Pour chaque pixel $\boldsymbol{k} = \begin{pmatrix} k \\ l \end{pmatrix}$ de l'image I_1 on définit un voisinage $\mathcal{W}(\boldsymbol{k})$ et on cherche à estimer la position $\boldsymbol{k} + \boldsymbol{u}(\boldsymbol{k})$ de ce voisinage dans l'image I_2 en minimisant le critère suivant

$$\boldsymbol{u}(\boldsymbol{k}) = \arg\min_{\boldsymbol{u}} \sum_{\boldsymbol{k}' \in \mathcal{W}(\boldsymbol{k})} \left(\tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}(\boldsymbol{k})) - I_1(\boldsymbol{k}') \right)^2 \tag{2.11}$$

On suppose alors que les déplacements à estimer sont petits, ou du moins, qu'on dispose d'une bonne initialisation $\boldsymbol{u}^{(0)}$, ce qui revient à poser

$$\boldsymbol{u}(\boldsymbol{k}) = \boldsymbol{u}^{(0)}(\boldsymbol{k}) + \boldsymbol{\delta u}(\boldsymbol{k}) \tag{2.12}$$

avec $\boldsymbol{\delta u}(\boldsymbol{k})$ petit. Dans cette hypothèse, on peut linéariser cette expression par un développement limité au premier ordre :

$$\boldsymbol{\delta u}(\boldsymbol{k}) = \arg\min_{\boldsymbol{\delta u}} \sum_{\boldsymbol{k}' \in \mathcal{W}(\boldsymbol{k})} \left(\tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k})) + \nabla \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}))^{\mathsf{T}} \cdot \boldsymbol{\delta u} - I_1(\boldsymbol{k}') \right)^2 \tag{2.13}$$

où $\nabla \tilde{I}_2 = \begin{pmatrix} \nabla_x \tilde{I}_2 \\ \nabla_y \tilde{I}_2 \end{pmatrix}$ est le gradient spatial de l'image cible (I_2) rééchantillonnée. La linéarisation au premier ordre étant une approximation, on utilise une méthode itérative. À chaque itération, on estime l'incrément $\boldsymbol{\delta u}$ à ajouter au flot $\boldsymbol{u}^{(0)}$ obtenu grâce aux itérations précédentes. En annulant la dérivée du critère (2.13) par rapport à $\boldsymbol{\delta u}$ on obtient :

$$\boxed{\mathscr{H} \cdot \boldsymbol{\delta u}(\boldsymbol{k}) = \sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \nabla \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k})) \left(I_1(\boldsymbol{k}') - \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k})) \right)} \tag{2.14}$$

avec

$$\mathscr{H} = \sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \nabla \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k})) \cdot \nabla \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}))^{\mathsf{T}} \tag{2.15}$$

La question de l'inversibilité de $\mathscr{H}$ est par exemple discutée dans SHI et collab. [1994] : si la fenêtre recouvre une région de l'image avec une texture suffisante, $\mathscr{H}$ est inversible. $\mathscr{H}$ est singulière si la fenêtre englobe une région trop homogène ou de contour rectiligne.

Le développement limité effectué ci-dessus est possible dans l'hypothèse d'un incrément $\boldsymbol{\delta u}$ petit. Une méthode permettant de gérer de plus grands déplacements, notamment proposée dans PyramLK (BOUGUET [2001]), consiste à utiliser l'approche *coarse-to-fine* déjà présentée en section 2.1.3, estimant une série de déplacements sur une pyramide d'images multi-échelle.

L'algorithme FOLKI

Une méthode locale dédiée au cas dense, FOLKI, a été développée à l'ONERA (LE BESNERAIS et CHAMPAGNAT [2005]). FOLKI utilise une pyramide multi-échelle et des itérations de type Lucas-Kanade à chaque niveau d'échelle, mais présente des gains calculatoires par une réorganisation des calculs dans le cas dense. Dans FOLKI on remplace $\boldsymbol{u}(\boldsymbol{k}) = \boldsymbol{u}^{(0)}(\boldsymbol{k}) + \boldsymbol{\delta u}$ par $\boldsymbol{u}(\boldsymbol{k}) = \boldsymbol{u}^{(0)}(\boldsymbol{k}') + (\boldsymbol{u}(\boldsymbol{k}) - \boldsymbol{u}^{(0)}(\boldsymbol{k}'))$ et on obtient :

$$\boldsymbol{u}(\boldsymbol{k}) = \arg\min_{\boldsymbol{u}} \sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \left(\tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}')) + \nabla \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}'))^{\mathsf{T}} \cdot (\boldsymbol{u}(\boldsymbol{k}) - \boldsymbol{u}^{(0)}(\boldsymbol{k}')) - I_1(\boldsymbol{k}') \right)^2 \tag{2.16}$$

Puis

$$\boxed{\left(\sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \nabla \tilde{I}_2^0(\boldsymbol{k}') \nabla \tilde{I}_2^0(\boldsymbol{k}')^{\mathsf{T}} \right) \cdot \boldsymbol{u}(\boldsymbol{k}) = \sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \nabla \tilde{I}_2^0(\boldsymbol{k}') \left(I_1(\boldsymbol{k}') - \tilde{I}_2^0(\boldsymbol{k}') + \nabla \tilde{I}_2^0(\boldsymbol{k}')^{\mathsf{T}} \cdot \boldsymbol{u}^{(0)}(\boldsymbol{k}') \right)} \tag{2.17}$$

avec $\tilde{I}_2^0(\boldsymbol{k}') = \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}'))$ et $\nabla \tilde{I}_2^0(\boldsymbol{k}') = \nabla \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}'))$. Cette astuce permet de réduire de manière importante le nombre d'interpolations et de calculs de gradients à réaliser. En effet, à chaque itération, $\tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}'))$ et $\nabla \tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}'))$ peuvent être calculés une seule fois pour tous les pixels, car il s'agit d'une interpolation de tout le champ image par un seul champ de déplacements. Au contraire, la quantité considérée dans l'algorithme de Lukas-Kanade : $\tilde{I}_2(\boldsymbol{k}' + \boldsymbol{u}^{(0)}(\boldsymbol{k}))$ fait intervenir une interpolation différente pour l'estimation en chaque pixel, puisqu'elle dépend de deux indices différents $\boldsymbol{k}'$ et $\boldsymbol{k}$.

La forme inverse

Pour les différentes méthodes locales présentées ci-dessus, il existe une forme inverse qui permet de réduire encore le temps de calcul (BAKER et MATTHEWS [2004]). Utiliser la

forme inverse revient à faire l'approximation suivante (avec les équations de l'algorithme de Lucas-Kanade) :

$$\underbrace{\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\left(\tilde{I}_2(\boldsymbol{k}'+\boldsymbol{u}^{(0)}(\boldsymbol{k})+\boldsymbol{\delta u}(\boldsymbol{k}))-I_1(\boldsymbol{k}')\right)^2}_{\text{Forme directe}} \simeq \underbrace{\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\left(\tilde{I}_2(\boldsymbol{k}'+\boldsymbol{u}^{(0)}(\boldsymbol{k}))-I_1(\boldsymbol{k}'-\boldsymbol{\delta u}(\boldsymbol{k}))\right)^2}_{\text{Forme inverse}} \tag{2.18}$$

Pour la forme inverse de FOLKI la quantité à minimiser est

$$\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\left(\tilde{I}_2(\boldsymbol{k}'+\boldsymbol{u}^{(0)}(\boldsymbol{k}'))-I_1(\boldsymbol{k}'-(\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}^{(0)}(\boldsymbol{k})))\right)^2 \tag{2.19}$$

Ou, après le développement limité :

$$\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\left(\tilde{I}_2(\boldsymbol{k}'+\boldsymbol{u}^{(0)}(\boldsymbol{k}'))-I_1(\boldsymbol{k}')+\nabla I_1(\boldsymbol{k}')^{\mathsf{T}}\cdot(\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}^{(0)}(\boldsymbol{k}'))\right)^2 \tag{2.20}$$

On obtient, en annulant la dérivée par rapport à $\boldsymbol{u}$:

$$\boxed{\boldsymbol{u}(\boldsymbol{k})=\mathscr{H}_1^{-1}\cdot\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\nabla I_1(\boldsymbol{k}')\left(I_1(\boldsymbol{k}')-\tilde{I}_2^0(\boldsymbol{k}')+\nabla I_1(\boldsymbol{k}')^{\mathsf{T}}\cdot\boldsymbol{u}^{(0)}(\boldsymbol{k}')\right)} \tag{2.21}$$

avec $\tilde{I}_2^0(\boldsymbol{k}')=\tilde{I}_2(\boldsymbol{k}'+\boldsymbol{u}^{(0)}(\boldsymbol{k}'))$ et $\mathscr{H}_1=\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\nabla I_1(\boldsymbol{k}')\nabla I_1(\boldsymbol{k}')^{\mathsf{T}}$. Une différence importante, entre le calcul du déplacement issu de la forme directe et de la forme inverse, réside dans le remplacement de $\nabla\tilde{I}_2^0(\boldsymbol{k}')$ par $\nabla I_1(\boldsymbol{k}')$. Dans la forme directe $\nabla\tilde{I}_2^0(\boldsymbol{k}')$ doit être recalculé après chaque mise à jour de $\boldsymbol{u}^{(0)}$, ce qui représente un nombre important d'interpolations et de calculs de gradients. Tandis que dans la forme inverse, $\nabla I_1(\boldsymbol{k}')$ peut n'être calculé qu'une seule fois pour toutes les itérations d'un même niveau de pyramide. En pratique, la précision de la forme inverse est comparable à celle de la forme directe, avec un gain calculatoire notable.

2.2.2 Les méthodes globales

L'article fondateur pour les méthodes globales d'estimation du flot optique est [HORN et SCHUNCK [1981]]. Les auteurs proposent de formuler le problème de l'estimation du flot optique comme la minimisation d'une énergie composée d'un terme d'attache aux données (version linéarisée de l'équation (2.2)) :

$$\mathscr{E}_{data}=\int_\Omega\left(I_2(\boldsymbol{x})-I_1(\boldsymbol{x})+\nabla I_2(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x})\right)^2\mathrm{d}\boldsymbol{x} \tag{2.22}$$

et d'un terme de régularisation (on favorise un flot lisse en minimisant la norme du gradient du flot) :

$$\mathscr{E}_{reg}=\int_\Omega\left(\|\nabla u(\boldsymbol{x})\|_2^2+\|\nabla v(\boldsymbol{x})\|_2^2\right)\mathrm{d}\boldsymbol{x} \tag{2.23}$$

Ω représente l'ensemble des sites de l'image (ici considérée continue).

L'énergie globale s'écrit :

$$\mathscr{E}=\mathscr{E}_{data}+\alpha\mathscr{E}_{reg} \tag{2.24}$$

Le paramètre réglable important dans ces méthodes est le paramètre de régularisation $\alpha>0$ qui règle le poids relatif entre la contrainte de régularisation (2.23) et l'attache aux données (2.22).

La contrainte de régularisation (2.23) revient à supposer que les trajectoires de points spatialement proches dans l'image diffèrent peu. Dans le cadre de scènes rigides ou partiellement rigides — *ie* composées d'objets rigides ou légèrement déformables, en mouvement dans un environnement rigide — cette hypothèse paraît raisonnable, sauf dans le cas singulier des discontinuités du mouvement aux frontières des objets. Cette contrainte permet de tenir compte d'une information portant sur la globalité du champ lors de l'estimation en chaque point. On résout ainsi la sous-détermination mentionnée plus tôt.

On cherche alors le champ $\boldsymbol{u}$ minimisant $\mathscr{E}$ de manière itérative. Avec cette méthode les régions non texturées se voient affectées d'un flot issu de la moyenne des flots avoisinants.

Cet algorithme présente deux inconvénients majeurs :

— Minimiser le carré de la norme du gradient du flot a pour effet de noyer les discontinuités du flot (contours, segmentation des objets).

— Le critère quadratique utilisé pour l'attache aux données est très sensible aux *outliers* évoqués en section 2.1.2.

Pour améliorer les performances de cette approche et préserver les discontinuités du flot des critères alternatifs à la pénalisation quadratique, issus de l'estimation robuste, ont été proposés. BROX et collab. [2004] utilisent ainsi une norme L_1 modifiée pour être dérivable en 0 :

$$\Psi(s^2) = \sqrt{s^2 + \epsilon^2} \text{ où } \epsilon = 0.001 \tag{2.25}$$

Les deux termes de l'énergie à minimiser s'écrivent alors :

$$\mathscr{E}_{data} = \int_\Omega \Psi\left(|I_2(\boldsymbol{x}+\boldsymbol{u}) - I_1(\boldsymbol{x})|^2 + \gamma \|\nabla I_2(\boldsymbol{x}+\boldsymbol{u}) - \nabla I_1(\boldsymbol{x})\|_2^2\right) \mathbf{d}\boldsymbol{x} \tag{2.26}$$

$$\mathscr{E}_{reg} = \int_\Omega \Psi\left(\|\nabla u(\boldsymbol{x})\|_2^2 + \|\nabla v(\boldsymbol{x})\|_2^2\right) \mathbf{d}\boldsymbol{x} \tag{2.27}$$

Cette pénalisation plus robuste autorise quelques écarts importants sur les différents critères, ce qui permet une meilleure restitution des discontinuités du flot, et une plus grande robustesse aux *outliers*. Il est également proposé d'ajouter, dans le terme d'attache aux données 2.26, un équivalent de l'équation (2.2) portant sur les gradients de l'image. Cela permet une plus grande robustesse aux changements d'intensité, les gradients de l'image y étant moins sensibles que l'image elle-même. Ces modifications mènent à de meilleurs résultats mais à une complexité calculatoire plus importante et donc des temps de calcul plus longs. La méthode implique aussi des paramètres supplémentaires à régler : le paramètre de seuil ϵ et le paramètre de poids relatif de l'énergie de recalage sur les gradients γ.

Une synthèse de nombreux travaux sur les méthodes globales

Depuis HORN et SCHUNCK [1981], de très nombreux travaux ont été réalisés sur l'approche variationnelle. L'article de SUN et collab. [2014] analyse et compare les différentes méthodes et astuces proposées jusqu'alors, et propose une synthèse des meilleures pratiques, ainsi qu'une implémentation de la méthode correspondante, basée sur celle de BROX et collab. [2004] et appelée *Classic + NL*. Les expériences de l'article montrent notamment une amélioration importante des résultats lors de l'application d'un filtre médian sur les flots intermédiaires, entre les itérations. Ceci améliore les résultats mais augmente l'énergie à minimiser, ce qui conduit les auteurs à proposer d'ajouter un terme non local à cette énergie, qui correspond à une opération de filtrage non linéaire, pour

remplacer ce filtre médian. D'autres bonnes pratiques sont soulignées, comme l'utilisation d'une pyramide asymétrique dans le cas d'images qui ne sont pas carrées, ce qui est fréquent.

La question des très grands mouvements

Nous avons vu que la résolution de l'équation du flot optique passait par une linéarisation impliquant une hypothèse de petits mouvements, et qu'une pyramide multi-échelle était utilisée pour permettre, malgré cela, la prise en compte de grands mouvements. Dans cette pyramide multi-échelle *coarse-to-fine*, les grands mouvements sont captés aux échelles grossières, où les structures spatiales fines sont perdues. Donc plus les mouvements à capter sont grands, plus les structures spatiales concernées doivent être spatialement étendues pour être correctement estimées. Autrement dit, cette stratégie échoue dans l'estimation du mouvement d'objets trop petits et rapides.

Si les méthodes que nous avons vu jusqu'ici fonctionnent bien sur le jeu de données Middlebury (BAKER et collab. [2007]), où l'amplitude des mouvements est raisonnable, elles sont mises en défaut dans le cas des mouvements beaucoup plus violents présents dans KITTI (MENZE et GEIGER [2015]) et dans certaines séquences de MPI Sintel (BUTLER et collab. [2012]). Dans ces jeux de données, les objets mobiles bougent souvent trop vite pour leur taille, et la stratégie *coarse-to-fine* usuelle est mise en échec.

Une nouvelle stratégie est alors proposée, pour capter ces grands mouvements, consistant à réaliser une première étape de mise en correspondance non dense et parfois grossière (on ne cherche pas une précision subpixellique à ce stade), afin de guider une estimation variationnelle du flot optique dans une seconde étape. Ainsi, la méthode LDOF (BROX et collab. [2009]) utilise une segmentation en régions et des descripteurs HOG (*Histogram of Oriented Gradients*) pour obtenir des correspondances, qui sont ensuite utilisées pour initialiser une méthode variationnelle. L'algorithme DeepFlow (WEINZAEPFEL et collab. [2013]) propose une nouvelle méthode de mise en correspondance quasi dense, DeepMatching (REVAUD et collab. [2016]), et ajoute un terme à l'énergie d'une méthode variationnelle pour tenir compte des correspondances obtenues. Le DeepMatching est réutilisé dans la méthode EpicFlow, par REVAUD et collab. [2015], dans laquelle la pyramide *coarse-to-fine* est complètement enlevée. Il ne reste que des itérations à pleine résolution lors de l'étape variationnelle, qui est initialisée par un flot obtenu par interpolation dense d'une mise en correspondance éparse. Cette interpolation dense tient compte, en plus des correspondances, d'une détection de contours de l'image source, afin de restituer au mieux les discontinuités du champ de mouvement. Enfin, l'algorithme FlowFileds (BAILER et collab. [2015]) reprend l'idée d'EpicFlow en proposant, pour remplacer le DeepMatching, une nouvelle méthode basée sur l'estimation d'un *dense approximate nearest neighbor field (ANNF)* suivi d'une détection et suppression d'*outliers*.

Ces dernières méthodes constituent l'état de l'art, avant l'arrivée des méthodes par apprentissage profond, sur les *benchmarks* Sintel et KITTI 2015, mais sont lentes : de l'ordre d'une vingtaine de secondes par paire d'images (BAILER et collab. [2015]; REVAUD et collab. [2015]; WEINZAEPFEL et collab. [2013]).

2.2.3 Discussion sur l'utilisation des approches locale et globale suivant le cadre applicatif

Le formalisme de l'approche globale a été beaucoup étudié et a donné lieu à des méthodes donnant des résultats très satisfaisants sur :

— les discontinuités spatiales du champ de mouvement, grâce à des critères robustes (BROX et collab. [2004]).

— les grands mouvements grâce à une étape préliminaire de mise en correspondance (BROX et collab. [2009]; WEINZAEPFEL et collab. [2013]).

De plus, leur régularisation globale permet une certaine robustesse au problème des zones peu texturées. Elles sont cependant coûteuses en temps de calcul, d'une part à cause de la minimisation d'une fonctionnelle globale, ce qui est lent à cause de la taille du système et l'est d'autant plus lorsqu'on utilise des critères robustes qui ne sont pas strictement convexes, d'autre part l'étape de mise en correspondance peut être chronophage. L'estimation du flot pour une paire d'image de Sintel (1024 × 436 pixels) prend 19 secondes avec DeepFlow (WEINZAEPFEL et collab. [2013]) et 16.4 secondes avec EpicFlow (REVAUD et collab. [2015]) sur un CPU à 3.6 GHz.

Les méthodes locales sont moins robustes au manque de texture d'une part, et donnent lieu à des effets de bavure — liés à l'hypothèse de flot constant dans la fenêtre — au niveau des discontinuités du champ de mouvement d'autre part. Elles peuvent cependant être très rapides. Pour différents algorithmes, PLYER et collab. [2016] comparent les temps de calculs du flot sur les jeux de données KITTI et Middlebury. L'algorithme FOLKI (local) est près de 100 fois plus rapide que l'algorithme global TV-L1 (ZACH et collab. [2007]), et 200 à 300 fois plus rapide que l'algorithme global de BROX et collab. [2004]. Dans son implémentation sur processeur graphique ou *graphics processing unit* (GPU), FOLKI présente un temps de calcul de 13 millisecondes pour une paire d'images 1920 × 1080 (sur une carte Nvidia Titan).

Moins répandues et moins étudiées que les méthodes globales dans le contexte de la vision par ordinateur, les méthodes locales trouvent néanmoins leur application dans deux contextes particuliers : la vision embarquée et la métrologie.

Dans le contexte de la vision embarquée, une exigence prioritaire est celle des coûts de calcul (temps et occupation mémoire). Or, d'une part, les méthodes locales sont nettement plus rapides que les méthodes globales. D'autre part, contrairement aux méthodes globales qui ne peuvent être que denses, les méthodes locales peuvent être utilisées pour calculer le flot en tout pixel (dense) mais aussi en une sous-sélection éparse de points, souvent issue d'une détection de points saillants. On peut donc alléger encore plus la chaîne de traitement lorsqu'il n'est pas nécessaire d'avoir un flot dense.
Note : Dans de nombreuses applications, il est toutefois important de considérer l'intérêt de disposer d'un flot dense, par exemple dans le cas de la détection d'objets mobiles, ou encore pour réaliser un recalage d'images non rigide. Cela justifie un nombre particulièrement importants de travaux sur les approches denses.

En métrologie, qu'on s'intéresse à la déformation d'un matériau ou au champ de vitesses d'un écoulement fluide, il est courant de pratiquer un ajout artificiel de texture sur le matériau ou une insémination de traceur dans le fluide. Les méthodes à fenêtre s'adaptent bien à ce contexte de la mesure par imagerie dans lequel la texture des images est contrôlée. Dans ce domaine, ces méthodes sont appréciées pour leur rapidité d'exécution et pour leur comportement et leur paramétrage intuitifs. De plus, les méthodes à fenêtre se prêtent bien à une analyse de l'incertitude résiduelle de l'estimation du déplacement, analyse qui est très importante pour les utilisateurs dans le contexte de la mesure. En mécanique des fluides, la technique de vélocimétrie par images de particules ou *Particle Image Velocimetry* (PIV) utilise des algorithmes à fenêtre pour estimer le déplacement, entre deux clichés, des traceurs inséminés dans le fluide. L'algorithme FOLKI présenté précédemment a été appliqué à cette tâche (CHAMPAGNAT et collab. [2011]).

2.3 Réseaux de neurones et apprentissage profond appliqués au flot optique

Depuis 2012, l'apprentissage profond révolutionne de nombreux domaines de la vision par ordinateur — ce qui nous intéresse ici — mais aussi la reconnaissance et la compréhension de la parole ou encore le traitement automatique du langage. Du côté de la vision par ordinateur, l'apprentissage profond a, dès 2012 (KRIZHEVSKY et collab. [2012]), établi un nouvel état de l'art en reconnaissance visuelle. Suite à cela, des méthodes basées sur l'apprentissage profond ont été proposées pour de nombreuses autres tâches de vision par ordinateur : super-résolution (TAO et collab. [2017]), estimation monoculaire de cartes de profondeur (EIGEN et collab. [2014]), estimation de disparité en stéréovision (ŽBONTAR et LECUN [2016]), *structure from motion* (UMMENHOFER et collab. [2017]), flot optique (DOSOVITSKIY et collab. [2015]).

2.3.1 Généralités sur l'apprentissage profond

Nous introduisons ici les notions qui nous paraissent essentielles pour aborder la suite du chapitre. Une description plus complète du domaine de l'apprentissage profond fait l'objet de nombreux ouvrages, dont certains sont disponibles en ligne (GOODFELLOW et collab. [2016]).

L'apprentissage profond est une branche de l'apprentissage automatique, basée sur l'utilisation de réseaux de neurones artificiels profonds inspirés du fonctionnement du cerveau humain. Un réseau de neurones profond est un ensemble de très nombreux neurones artificiels, réalisant des opérations élémentaires, organisé en de multiples couches successives. L'opération réalisée par un neurone est basiquement une somme pondérée de ses entrées, dont les pondérations sont des paramètres qu'il faudra apprendre lors d'une phase d'entraînement, suivie d'une fonction d'activation non linéaire.

Dans le contexte de la vision par ordinateur, on utilise plus particulièrement des réseaux de neurones convolutifs ou *Convolutional Neural Networks (CNNs)*. Dans ces réseaux on répète en tout point du champ image les mêmes opérations de convolution, qui sont des opérations locales considérant un voisinage spatial limité autour de chaque pixel. Le résultat, au pixel de coordonnées (i, j), de la convolution discrète d'une image I par un noyau de convolution carré $\mathcal{N}$ de taille $K \times K$ s'écrit :

$$(I * \mathcal{N})(i,j) = \sum_{m=-K'}^{K'} \sum_{n=-K'}^{K'} I(i-m, j-n)\mathcal{N}(m,n) \tag{2.28}$$

avec $K = 2K' + 1$. Les valeurs des noyaux sont des paramètres du réseaux, qui seront appris pendant l'entraînement.

En général, les convolutions par plusieurs noyaux appris sont appliquées au tenseur d'entrée, et les cartes d'activations issues de toutes les convolutions sont concaténées. Ceci forme ce qu'on appelle une "couche" de convolution. Un CNN en comporte plusieurs, le tenseur de sortie d'une couche étant le tenseur d'entrée de la couche suivante. On désigne par "nombre de canaux de sortie" C_{out} d'une couche convolutive le nombre de cartes d'activations calculées, *ie* le nombre de canaux de son tenseur de sortie, correspondant donc au nombre de noyaux de convolution de la couche. Lorsqu'une couche a plusieurs canaux d'entrée C_{in}, ce qui arrive très souvent puisque c'est le cas des images en couleurs ou encore des tenseurs intermédiaires entre deux couches convolutives, chaque noyau de convolution a également C_{in} canaux, ce qui revient à avoir C_{in} noyaux 2D, et le

canal $c' \in [0, C_{\text{out}}]$ de la sortie de la couche est calculé comme suit :

$$\mathscr{F}_i(c') = \sum_{c=1}^{C_{\text{in}}} \mathscr{W}(c', c) * \mathscr{F}_{i-1}(c) \tag{2.29}$$

où $*$ représente le produit de convolution, les $\mathscr{W}(c', c)$ sont des noyaux de convolution 2D et $\mathscr{F}_{i-1}$ et $\mathscr{F}_i$ sont respectivement l'entrée et la sortie de la i-ème couche convolutive. Une couche convolutive avec C_{in} canaux d'entrée, C_{out} canaux de sortie, et des noyaux de convolution carrés de taille $K \times K$, représente donc $K^2 \times C_{\text{in}} \times C_{\text{out}}$ poids de convolution, auxquels peuvent s'ajouter C_{out} paramètres appelés "biais", le c'-ième biais étant ajouté à $\mathscr{F}_i(c')$ après le calcul de la convolution. D'une couche à la suivante, il est courant de réduire la dimension spatiale et d'augmenter le nombre de canaux (SIMONYAN et ZISSERMAN [2014]).

Ces réseaux de neurones sont généralement entraînés de bout en bout, on parle d'apprentissage *end-to-end* : on donne des images en entrée du réseau et on en pénalise la sortie au moyen d'une fonction de coût — censée être minimale lorsque le réseau réalise correctement la tâche souhaitée — on met ensuite à jour les paramètres du réseau afin de minimiser cette fonction de coût. Pour cela, les gradients de la fonction de coût par rapport à chaque paramètre sont calculés par rétropropagation à travers le réseau, et les paramètres sont mis à jour de manière itérative, afin de réduire l'erreur calculée par la fonction de coût, par descente de gradient stochastique. On appelle "taux d'apprentissage" le pas de la descente de gradient. Ceci nécessite un jeu de données d'entraînement nombreuses et variées, sans quoi le réseau risquerait de sur-apprendre des détails liés aux exemples d'entraînement au lieu d'apprendre la tâche souhaitée et de pouvoir généraliser à d'autres données.

Si les données d'entraînement sont annotées (*ie* on dispose de la vérité terrain pour la tâche souhaitée), on parle d'apprentissage supervisé. La fonction de coût peut alors être une distance, dépendant de la tâche d'intérêt, entre la prédiction et la vérité terrain. Dans le cas contraire, on parle d'apprentissage non supervisé, ou auto-supervisé, il faut alors concevoir une fonction de coût pénalisant la sortie du réseau d'une autre manière, comme nous le verrons en 2.3.4, dans le cas du flot optique.

2.3.2 Les fondements de l'estimation de flot optique par CNN

Une approche par apprentissage profond pour l'estimation de flot optique est proposée pour la première fois par DOSOVITSKIY et collab. [2015]. Les auteurs posent alors les bases d'une nouvelle famille de méthodes en proposant une architecture de réseau de neurones (déclinée en deux variantes : FlowNetSimple et FlowNetCorr), un jeu de données annoté — FlyingChairs — pour l'entraînement, et une fonction de coût multi-échelle. Contrairement aux méthodes basées modèle qui cherchent à estimer un flot optique à partir de deux images en minimisant un critère sur les intensités, cette nouvelle approche consiste à apprendre la tâche d'estimation du flot optique, par supervision à partir d'un volume important de données annotées. Cela se fait en optimisant itérativement, à mesure qu'on "nourrit" le réseau avec les données d'entraînement, les poids — ou "paramètres" — du réseau de sorte à minimiser la fonction de coût.

Fonction de coût

Les réseaux proposés sont entraînés de manière supervisée en cherchant à minimiser une fonction de coût basée sur l'erreur *endpoint* par rapport à la vérité terrain du

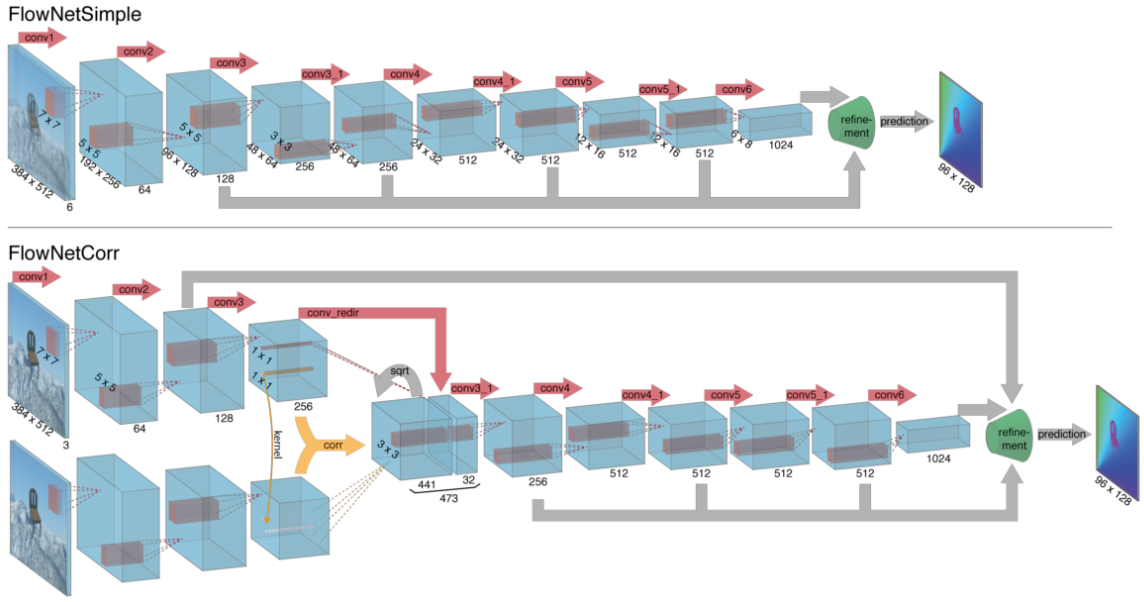

FIGURE 2.7 – Schémas de FlowNetSimple et FlowNetCorr. Le détail du décodeur ("refinement") est donné sur la figure 2.8. Source : DOSOVITSKIY et collab. [2015].

flot optique. Plus précisément, les réseaux fournissent une estimation du flot à plusieurs échelles et la fonction de coût est la somme pondérée des erreurs *endpoint* moyennes aux L différentes échelles : on cherche donc les paramètres Θ du réseau minimisant

$$\mathscr{L}(\Theta) = \sum_{l=l_0}^{L} \alpha_l \sum_{\boldsymbol{x}} \left\| \boldsymbol{u}_{\Theta}^{(l)}(\boldsymbol{x}) - \boldsymbol{u}_{\mathrm{VT}}^{(l)}(\boldsymbol{x}) \right\| \tag{2.30}$$

où $\|\cdot\|$ représente la norme euclidienne, $\boldsymbol{u}_{\Theta}^{(l)}(\boldsymbol{x})$ le flot estimé et $\boldsymbol{u}_{\mathrm{VT}}^{(l)}(\boldsymbol{x})$ la vérité terrain du flot optique, au pixel de coordonnées $\boldsymbol{x}$.

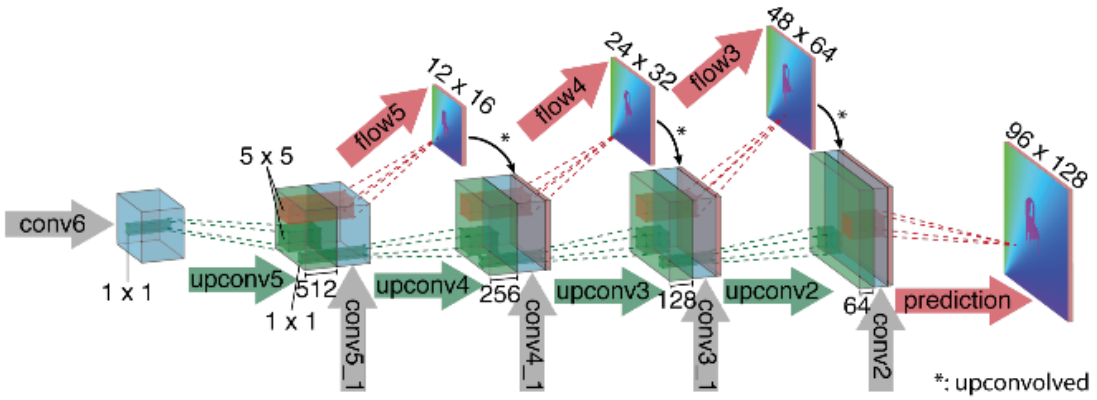

FIGURE 2.8 – Schéma du décodeur de FlowNetSimple et FlowNetCorr. Source : DOSOVITSKIY et collab. [2015].

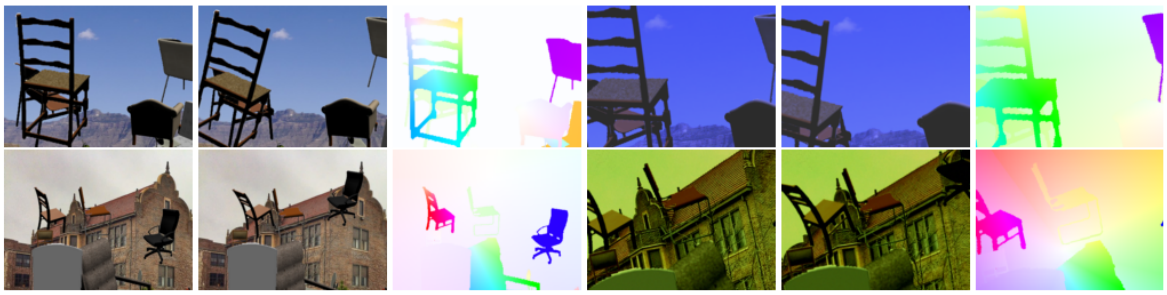

FIGURE 2.9 – Exemples issus du jeu de données FlyingChairs. Source : DOSOVITSKIY et collab. [2015].

Architectures FlowNetSimple et FlowNetCorr

Présentation générale. Les deux architectures proposées, figure 2.7, sont de type encodeur-décodeur, inspirées de l'architecture U-Net (RONNEBERGER et collab. [2015]). Par rapport au réseau U-Net, une convolution supplémentaire est utilisée à chaque étage du décodeur pour estimer un flot à chaque résolution intermédiaire. La fonction de coût multi-échelle de l'équation 2.30 pénalise le flot final et les flots intermédiaires. FlowNetSimple concatène les deux images d'entrées, de sorte que l'entrée de l'encodeur soit un tenseur à 2 canaux pour des images en niveaux de gris, ou 6 canaux pour des images en couleurs. FlowNetCorr est un réseau siamois : il possède deux voies d'entrée distinctes mais effectuant les mêmes opérations (les poids des convolutions sont partagés entre les deux voies). Chacune des deux images d'entrée subit donc les mêmes traitements indépendamment puis, au milieu de l'encodeur, les activations des deux voies sont fusionnées au moyen d'une "couche de corrélation" : la suite de l'encodeur est mono-voie et prend en entrée la concaténation du résultat de la corrélation, avec les cartes d'activation d'une des voies avant la corrélation.

Description de la couche de corrélation. Notons F_1 (respectivement F_2) les activations issues de l'image source I_1 (respectivement l'image cible I_2), juste avant la couche de corrélation. Cette couche calcule une corrélation entre ces activations, autrement dit une corrélation entre descripteurs appris. Pour une position $\boldsymbol{x}_1$ dans I_1 et une position $\boldsymbol{x}_2$ dans I_2, la corrélation entre descripteurs appris consiste à calculer le produit scalaire entre $F_1(\boldsymbol{x}_1)$ et $F_2(\boldsymbol{x}_2)$. La couche de corrélation consiste à réaliser cette opération pour tout x_1 appartenant au support image et pour tout x_2 dans un voisinage autour de x_1. Un déplacement maximal de $d = 20$ pixels autour de x_1 est considéré.

Résolution de l'estimation. L'encodeur présente 6 étages de réduction de la dimension spatiale, le décodeur ne réaugmente celle-ci que 4 fois, d'un facteur deux à chaque fois. L'estimation finale est donc quatre fois moins résolue que l'image d'entrée, sur chaque dimension spatiale. Sauf mention du contraire, toutes les méthodes présentées dans la suite de ce chapitre suivent le même principe : l'estimation s'arrête au quart de la résolution d'entrée puis une interpolation bilinéaire permet un sur-échantillonnage d'un facteur 4 pour atteindre la pleine résolution. D'après les auteurs, ajouter des niveaux supplémentaires au décodeur pour obtenir une estimation à pleine résolution n'améliore pas suffisamment les résultats pour justifier les coûts calculatoires engendrés, bien plus élevés que ceux d'une interpolation bilinéaire.

Entraînement

La base d'entraînement FlyingChairs est composée de plus de 22 000 paires d'images synthétiques accompagnées de la vérité terrain du flot optique. Ces images sont générées en superposant des images de chaises sur des fonds divers, et en appliquant aux chaises et au fond des transformations affines 2D différentes. La figure 2.9 montre des exemples issus de cette base d'entraînement.

Diverses augmentations de données sont pratiquées durant l'entraînement, pour limiter le phénomène de sur-apprentissage. Certaines transformations sont géométriques (translations, rotations et homothéties), d'autres sont photométriques (ajout de bruit gaussien, modifications aléatoires de la luminosité, du gamma, du contraste et de la couleur). Les expériences présentées par DOSOVITSKIY et collab. [2015] montrent l'importance de ces augmentations de données : sans elles les performances obtenues sur Sintel sont grandement dégradées (différence d'environ 25% sur l'EPE moyenne) à l'issue de l'apprentissage.

Après cet entraînement sur FlyingChairs, une phase de *finetuning* est réalisée sur l'ensemble d'entraînement de Sintel. Le *finetuning* consiste à réaliser une deuxième phase d'entraînement sur d'autres données, en repartant du modèle pré-entraîné issu de la première phase d'entraînement (ici, sur FlyingChairs), et en commençant avec un taux d'apprentissage plus faible que lors de la première phase. Les expériences proposées par les auteurs montrent l'intérêt de cet entraînement en deux phases :

— Entraîner directement sur Sintel, sans pré-entraînement sur FlyingChairs, donne de moins bons résultats sur l'ensemble d'évaluation de Sintel (différence d'environ 15% sur l'EPE moyenne).

— Le *finetuning* sur l'ensemble d'entraînement de Sintel améliore les performances sur l'ensemble d'évaluation de Sintel mais aussi sur celui de KITTI 2012.

Résultats de FlowNetS et FlowNetC

Les résultats obtenus ne sont pas à l'état de l'art sur les *benchmarks* (MPI Sintel et KITTI 2012) mais sont suffisamment bons pour se classer devant la méthode LDOF (BROX et collab. [2009]) sur Sintel et KITTI, malgré la différence notable entre ces jeux de données et les exemples d'entraînement de FlyingChairs. Les auteurs remarquent notamment que FlowNetS et FlowNetC, bien que moins performants que la méthode variationnelle EpicFlow (REVAUD et collab. [2015]) au sens de l'EPE, donnent une estimation plus satisfaisante sur les détails spatiaux fins, et certains objets petits et rapides.

FlyingThings3D

Une nouvelle base d'apprentissage, annotée pour l'estimation de flot optique, est ensuite proposée par MAYER et collab. [2016] : également simulée, cette base "FlyingThings3D", plus complexe, met en scène des modèles 3D d'objets se déplaçant dans une scène elle aussi à trois dimensions. Ce nouveau jeu de données est composé de plus de 2500 séquences de 9 à 10 images, annotées des flots optiques aller et retour, de la disparité, de la segmentation des objets et des discontinuités du mouvement. Les mouvements y sont plus complexes et violents que dans FlyingChairs, et les objets mobiles ne se limitent pas à des chaises. La figure 2.10 montre quelques exemples issus de FlyingThings3D (image source et VT du flot optique).

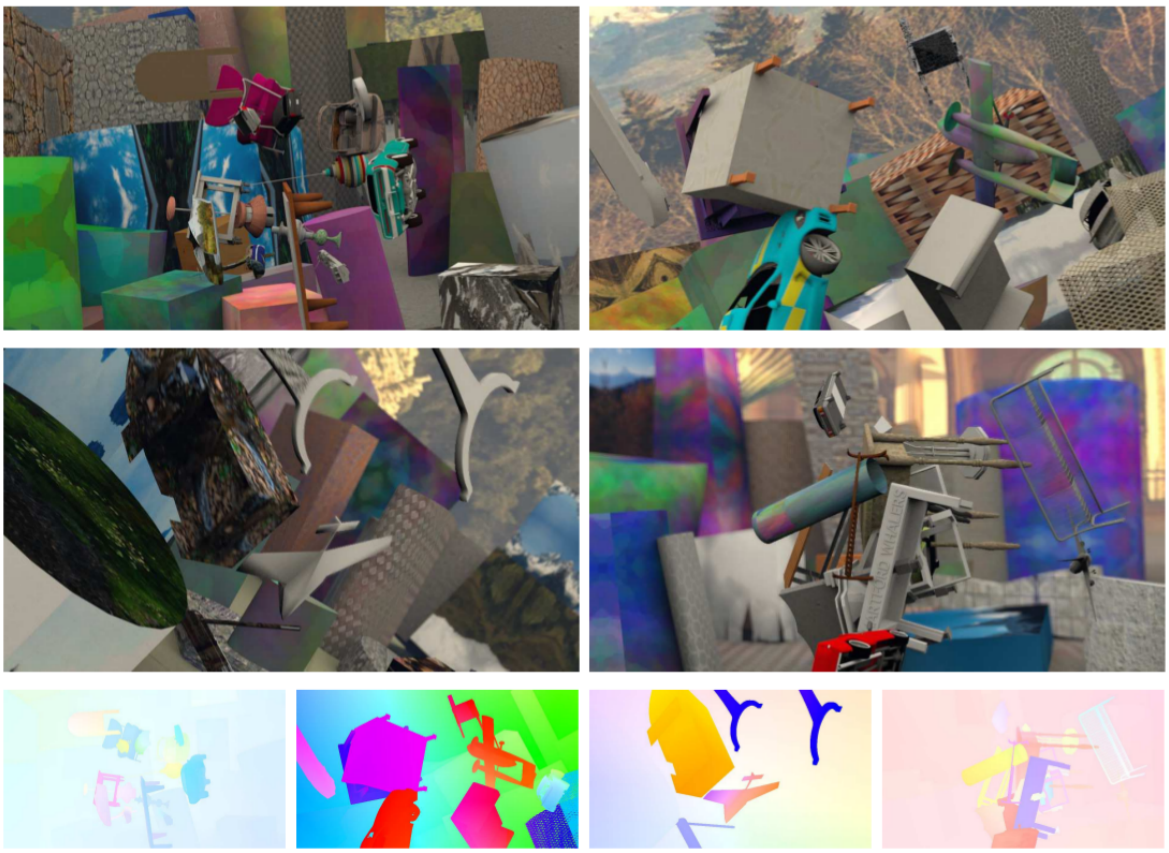

FIGURE 2.10 – Exemples issus du jeu de données FlyingThings3D. Source : MAYER et collab. [2016].

FlowNet2

Les premiers résultats obtenus par apprentissage profond et atteignant l'état de l'art sont obtenus par la méthode FlowNet2 (ILG et collab. [2017]). Cette méthode met en cascade différentes architectures de type FlowNetCorr et FlowNetSimple, et les entraîne sur FlyingThings3D après un pré-entraînement sur FlyingChairs. Plus précisément, un sous-ensemble de FlyingThings3D est utilisé, nommé "FlyingThings3D *subset*", afin d'ignorer les séquences de FlyingThings3D jugées trop complexes par les auteurs. Ce "subset" comporte 2183 séquences d'entraînement (soit 19636 paires d'images d'entraînement) et 425 séquences de validation.

Les auteurs étudient différents enchaînements de blocs FlowNetS/FlowNetC pour leur méta-architecture. Ils montrent que FlowNetC donne de meilleurs résultats que FlowNetS, et choisissent FlowNetC comme première brique de leur architecture. Les autres briques sont des instances de FlowNetS. Après chaque bloc, l'image cible est rééchantillonnée (*warping*) de manière à compenser le flot déjà estimé. Ainsi, le bloc suivant doit estimer le flot résiduel, comme s'il s'agissait d'une itération d'une méthode classique d'estimation résiduelle de flot optique, comme décrit par BROX et collab. [2004]. Les différents réseaux mis en cascade sont entraînés les uns après les autres. Enfin, un réseau supplémentaire est spécialisé dans les petits mouvements (entraîné sur une version de FlyingChairs à petits mouvements), puis un réseau de fusion estime le flot final à partir du flot de la branche principale et de celui de la branche des petits mouvements.

La stratégie d'entraînement

ILG et collab. [2017] évaluent également les performances des architectures de base (FlowNetS et FlowNetC) en fonction des données d'apprentissage présentées :

— FlyingChairs puis FlyingThings ;

— FlyingChairs uniquement ;

— FlyingThings uniquement ;

— Mélange aléatoire des deux bases.

Entraîner uniquement sur FlyingThings3D est sous-optimal, ceci donne de moins bons résultats (sur Sintel Clean) qu'en entraînant uniquement sur FlyingChairs. L'entraînement sur FlyingChairs puis sur FlyingThings3D donne de meilleurs résultats que l'entraînement sur le mélange des deux bases ou sur FlyingChairs uniquement, à nombre d'itérations total constant. Ils montrent ainsi que les résultats obtenus sont meilleurs en entraînant sur des données simples d'abord puis plus compliquées ensuite. Cette conclusion va imposer une stratégie d'entraînement reprise par de nombreux travaux sur l'estimation de flot optique par apprentissage profond (HUR et ROTH [2019]; NEORAL et collab. [2018]; SUN et collab. [2018]) : FlyingChairs → FlyingThings3D (*subset*), puis un éventuel *finetuning* sur des données proches de l'application ciblée, notamment sur les ensembles d'entraînements des *benchmarks* avant l'évaluation. Nous donnons plus de détails sur cette stratégie d'entraînement dans l'annexe A. Cette stratégie d'entraînement, consistant à présenter d'abord des exemples simples puis des exemples plus compliqués, est appelée *curriculum learning* (BENGIO et collab. [2009]).

2.3.3 Tirer profit des bonnes pratiques utilisées dans les méthodes classiques (basées modèle)

FlowNet2 (ILG et collab. [2017]) est une méthode très performante, établissant le nouvel état de l'art au moment de sa publication, mais à l'architecture très lourde. De nombreux travaux ont suivi, pour proposer des architectures alliant performance et légèreté en s'inspirant de techniques d'estimation de flot optique plus classiques. Ainsi, en utilisant une pyramide d'échelle et en estimant le flot résiduel à chaque échelle, RANJAN et BLACK [2017] obtiennent une architecture, nommée SpyNet, plus légère que FlowNet (DOSOVITSKIY et collab. [2015]) mais présentant des résultats proches. Les architectures PWC-Net (SUN et collab. [2018]) et LiteFlowNet (HUI et collab. [2018]) reprennent cette stratégie multi-échelle, et réalisent à chaque échelle une estimation du flot en s'appuyant sur le calcul d'une corrélation entre les cartes d'activations issues des deux images d'entrée. Ces deux architectures présentent des résultats à l'état de l'art tout en restant nettement plus légères que FlowNet2. Une avancée supplémentaire en termes de légèreté est proposée par HUR et ROTH [2019] qui proposent une variante des architectures FlowNet et PWC-Net où les mêmes poids sont utilisés pour l'estimation des flots à tous les niveaux d'échelle. La figure 2.11 résume les performances et nombres de paramètres de ces méthodes.

PWC-Net

Nous décrivons ici plus en détails l'architecture PWC-Net, proposée par SUN et collab. [2018], servant de base à de nombreux autres travaux (BAR-HAIM et WOLF [2020]; HUR et ROTH [2019]; LIU et collab. [2019]; NEORAL et collab. [2018]; REN et collab. [2019]). En

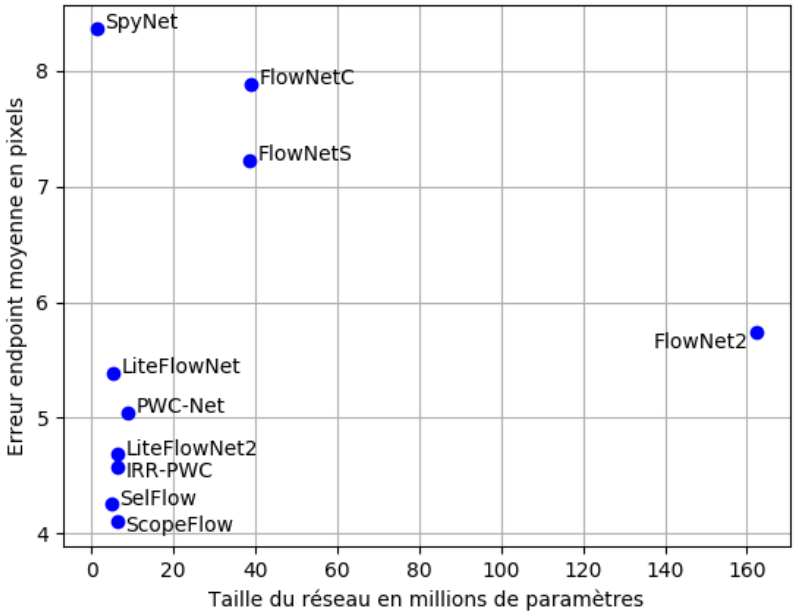

FIGURE 2.11 – Comparaison des méthodes d'estimation de flot optique par apprentissage profond, en termes de nombre de paramètres appris et d'erreur *endpoint* moyenne sur le *benchmark* MPI Sintel Final.

plus d'être plus légère et performante que FlowNet2, PWC-Net est également plus simple à entraîner, ne nécessitant pas l'entraînement séparé de différents blocs.

PWC-Net fait intervenir des opérations apprises (convolutions par des noyaux aux poids appris) et des opérations déterministes inspirées des techniques couramment utilisées dans des méthodes plus classiques d'estimation de flot optique. On retrouve notamment une stratégie multi-échelle, avec estimation à chaque étape du flot résiduel après un pré-recalage (*warping*) réalisé à l'aide du flot estimé à l'échelle précédente. PWC-Net utilise également une couche de corrélation, que l'on peut assimiler au calcul d'un critère de similarité dans l'hypothèse de la conservation de l'intensité lumineuse le long des trajectoires. Cette couche de corrélation est la même que celle utilisée dans FlowNetCorr, et décrite en section 2.3.2, mais en considérant un déplacement maximal de $d = 4$ pixels au lieu de $d = 20$ pixels pour FlowNetCorr. Ce déplacement maximal est choisi plus petit en raison du mécanisme multi-échelle — avec pré-recalage à chaque échelle — de PWC-Net qui permet l'estimation de grands mouvements.

Un même encodeur est utilisé pour générer des "descripteurs appris" multi-échelles à partir de chacune des deux images d'entrée. Pour chacune des deux images on obtient une pyramide de descripteurs à différentes échelles, la taille de chacune des dimensions spatiales est divisée par deux d'un niveau d'échelle au suivant. Sur le schéma de la figure 2.12, $P^i(I_1)$ représente le jeu de descripteurs à l'échelle i pour l'image source I_1. À chaque niveau d'échelle, en commençant par le plus grossier, on calcule un "cost-volume" par corrélation des descripteurs appris issus des deux images. Afin de limiter le nombre de calculs et la taille de ce "cost-volume", on considère un déplacement maximal de 4 pixels lors du calcul de la corrélation. Un réseau de neurone convolutif "Optical Flow Decoder", composé d'un enchaînement de 5 couches de convolution densément connectées (HUANG et collab. [2017]), prend en entrée le résultat de la corrélation, les

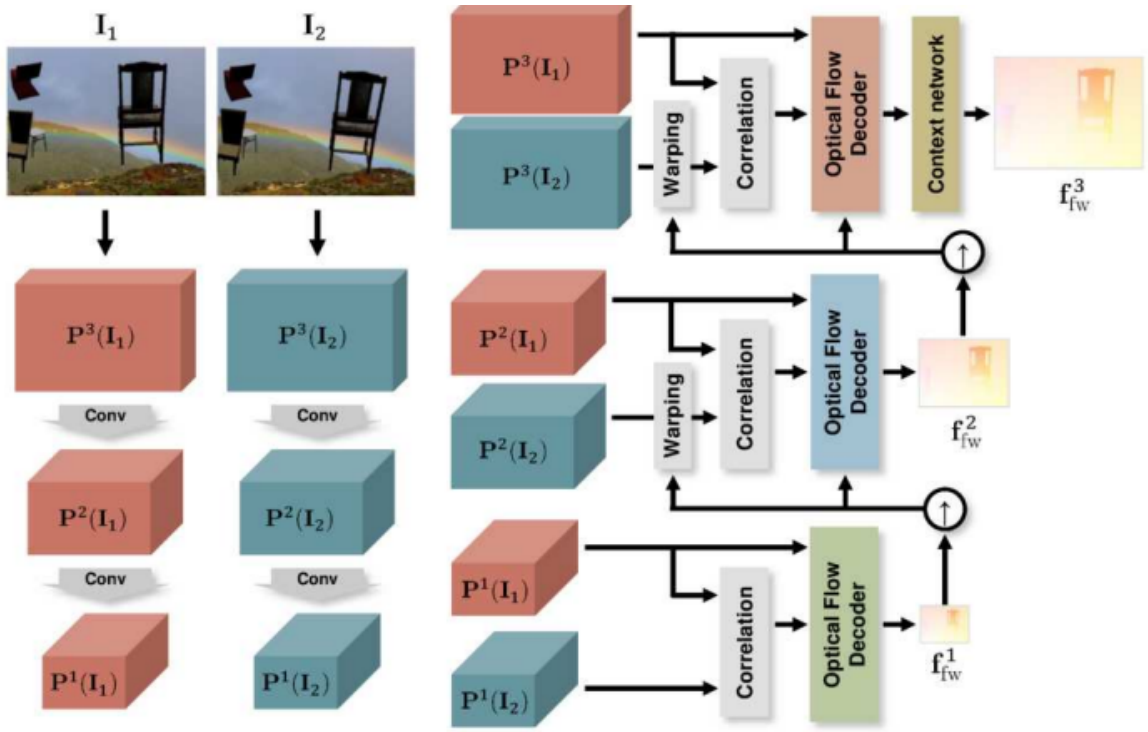

FIGURE 2.12 – Schéma de principe de PWC-Net. Seuls trois niveaux d'échelle ont été représentés pour plus de clarté. ↑ : sur-échantillonnage d'un facteur 2. Source : HUR et ROTH [2019].

descripteurs de l'image source (à l'échelle courante) et le flot estimé à l'échelle précédente (à l'échelle la plus grossière ce flot est nul), et donne en sortie une estimation du flot à l'échelle courante. Ce flot sera utilisé à l'échelle suivante, après un sur-échantillonnage aux plus proches voisins, pour effectuer un pré-recalage (*warping*) des descripteurs de l'image cible I_2 avant le calcul de la corrélation, et pour être donné en entrée d'un nouvel "Optical Flow Decoder".

Ce mécanisme multi-échelle s'arrête quand on arrive au quart de la résolution des images d'entrée. Le flot obtenu est alors passé dans un "context network" opérant des convolutions dilatées apprises (YU et collab. [2017]), puis interpolé bilinéairement à pleine résolution. Pour les auteurs ce "context network" rappelle une étape de post-traitement comme l'application du filtre médian dans la méthode proposée par SUN et collab. [2014], comme évoqué en section 2.2.2.

Cette architecture est entraînée suivant la stratégie de présentation des données d'entraînement proposée par ILG et collab. [2017], et décrite en section 2.3.2 : FlyingChairs puis FlyingThings, puis un éventuel *finetuning* sur Sintel ou KITTI.

IRR-PWC

HUR et ROTH [2019] proposent une variante de PWC-Net, IRR-PWC, réalisant une estimation résiduelle itérative en itérant sur les mêmes poids appris. Le réseau d'estimation de flot optique ("Optical Flow Decoder") est donc le même pour toutes les échelles de la pyramide, et des connexions résiduelles sont ajoutées. À chaque itération, le réseau réalise l'estimation résiduelle pour obtenir le flot à la nouvelle échelle, à partir d'un flot plus grossier, sur-échantillonné aux plus proches voisins, voir figure 2.13. Ce partage des mêmes poids appris pour l'estimation du flot aux différentes échelles permet une réduction de la taille du modèle, tout en donnant de meilleures performances, comme le

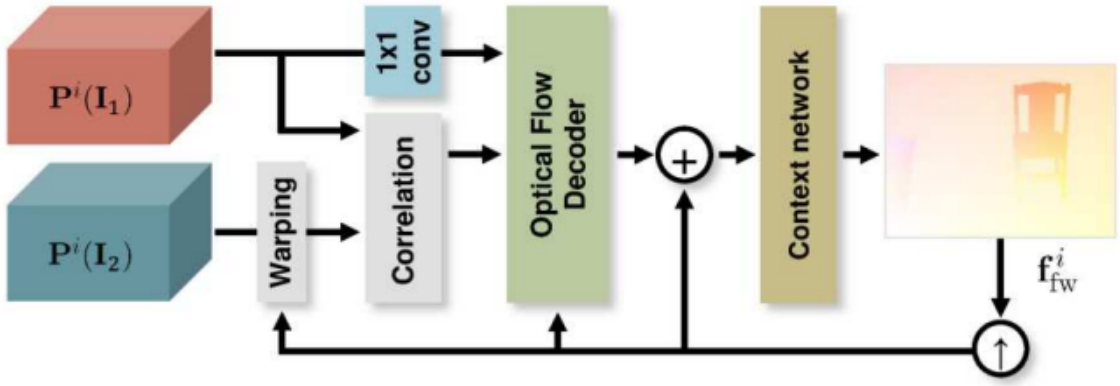

FIGURE 2.13 – Schéma de principe d'une variante de PWC-Net utilisant le principe d'itération sur les mêmes poids ("iterative residual refinement"), proposé dans IRR-PWC. ↑ : sur-échantillonnage d'un facteur 2. Source : HUR et ROTH [2019].

montrent les évaluations menées par les auteurs sur Sintel dans leur premier tableau.

D'autres modifications distinguent PWC-Net et IRR-PWC. Premièrement, une estimation des cartes d'occultations est réalisée en plus de l'estimation du flot. Pour réaliser un entraînement supervisé, les auteurs proposent une version de FlyingChairs, appelée FlyingChairsOcc, annotée des cartes d'occultations. Le *subset* de FlyingThings3D est déjà annoté de la vérité terrain des cartes d'occultations. Deuxièmement, le flot est estimé dans les deux sens et la fonction de coût pénalise ces deux estimations. Le même réseau est utilisé deux fois pour chaque paire d'images, en inversant l'ordre des deux images, pour estimer le flot aller et le flot retour. Pour superviser cet entraînement, FlyingChairOcc est annoté du flot et des occultations dans les deux sens pour chaque paire d'images. FlyingThings3D propose également la vérité terrain de ces quantités dans les deux sens. Troisièmement, un module de *refinement* affine l'estimation grâce à l'application de filtres de convolution localement générés par un réseau à partir des descripteurs de l'image source, du flot à affiner et de l'erreur photométrique associée. Enfin, si une interpolation bilinéaire est effectuée pour passer à pleine résolution à partir du flot estimé par le réseau au quart de la résolution d'entrée, les occultations subissent un autre traitement : un réseau réalise leur sur-échantillonnage, en tenant compte de l'estimation aller-retour du flot.

À sa publication, IRR-PWC établit le nouvel état de l'art au sein des méthodes biframe, sur MPI Sintel et KITTI 2015. BAR-HAIM et WOLF [2020] reprennent cette architecture, et améliorent les résultats en changeant le protocole d'apprentissage. Ce nouveau protocole est établi à l'issue d'une étude sur l'impact des transformations géométriques utilisées comme augmentation de données. La méthode obtenue avec ce nouvel entraînement est appelée ScopeFlow.

2.3.4 Apprentissage non supervisé

Les méthodes présentées ci-dessus sont supervisées : les réseaux sont entraînés en minimisant une distance entre leur estimation et la vérité terrain. D'autres travaux proposent un entraînement non supervisé en minimisant une fonction de coût inspirée des fonctionnelles utilisées dans les méthodes variationnelles de flot optique, comme celle de l'équation (2.24). Une telle fonction de coût, pour l'apprentissage non supervisé du

flot, fait donc intervenir un terme d'attache aux données (2.26) mesurant l'erreur photométrique entre l'image source et l'image cible rééchantillonnée aux positions données par le flot estimé, et un terme de régularisation spatiale. Ainsi, JASON et collab. [2016] et REN et collab. [2017] entraînent FlowNetS sur FlyingChairs et KITTI 2012, sans utiliser la vérité terrain. Les résultats présentés dans ces travaux montrent que, à données d'entraînement et architectures équivalentes, si les données sont densément annotées (c'est le cas de FlyingChairs), l'entraînement supervisé donne des résultats nettement meilleurs que l'entraînement non supervisé. Cependant, sur KITTI, seule une paire d'images par séquence est annotée, et cette annotation est spatialement éparse. L'entraînement non supervisé permet d'utiliser les paires d'images non annotées de KITTI, ainsi que la totalité des pixels, ce qui augmente considérablement le nombre d'exemples d'apprentissages disponibles sur ce jeu de données. Dans ce cas, la stratégie non supervisée est meilleure, sauf dans les zones occultées.

Dans le terme d'attache aux données de la fonction de coût non supervisée, les pixels occultés dans l'image cible posent problème : leur erreur sera haute même si l'estimation est bonne, et l'erreur photométrique minimale peut tout à fait correspondre à une estimation complètement erronée, ce qui peut induire le réseau en erreur et perturber l'entraînement. MEISTER et collab. [2018] proposent d'ignorer ces pixels dans le calcul de la fonction de coût. Pour cela, le flot est estimé dans les deux sens, et un critère de cohérence entre les flots aller et retour permet de détecter les zones d'occultations. Afin d'éviter de bloquer l'optimisation du réseau dans une situation où tous les pixels seraient considérés comme occultés — et donc ignorés — un terme est ajouté à la fonction de coût pour pénaliser le nombre de pixels considérés comme occultés. Ils proposent également d'utiliser la transformée census (ZABIH et WOODFILL [1994]) afin d'être plus robuste aux changements d'intensités lumineuses avant le calcul de l'erreur photométrique. Ils entraînent ainsi l'architecture FlowNetC et une version simplifiée de FlowNet2 (sans la branche dédiée aux petits mouvements). Un pré-entraînement est réalisé sur le jeu de données synthétiques de navigation routière SYNTHIA (ROS et collab. [2016]) puis la suite de l'entraînement est faite sur les données non annotées de KITTI ou sur Cityscapes (CORDTS et collab. [2016]), un autre jeu de données réelles de navigation routière mais initialement dédié à la segmentation sémantique. L'ensemble de ces modifications mène à une amélioration notable des résultats par rapport aux méthodes non supervisées précédentes. Leurs expériences montrent en particulier une amélioration très importante grâce à la transformée census. Le fait d'ignorer les zones occultées dans le calcul de la fonction de coût leur permet d'améliorer les résultats dans les zones occultées de KITTI.

JANAI et collab. [2018] proposent une méthode non supervisée basée sur PWC-Net, ignorant également les pixels occultés dans la fonction de coût mais au moyen d'un masque d'occultations appris. Leur architecture est une variante de PWC-Net considérant 3 images (I_{t-1}, I_t et I_{t+1}) et estimant le flot de t vers $t-1$, le flot de t vers $t+1$, et une carte d'occultations (à deux canaux) indiquant pour chaque pixel de I_t s'il est visible à $t-1$, à $t+1$ ou aux deux. Ils ajoutent à la fonction de coût un terme de vitesse constante, favorisant les flots de t vers $t-1$ et de t vers $t+1$ à être l'opposé l'un de l'autre.

Si l'idée d'ignorer les pixels occultés dans la fonction de coût permet d'éviter d'induire l'estimateur en erreur pendant l'entraînement, cela ne permet pas pour autant de lui apprendre à prédire le mouvement pour les pixels occultés. Le mouvement de ces pixels ne sera en fait jamais pénalisé. La méthode SelFlow (LIU et collab. [2019]) propose un entraînement permettant d'apprendre au réseau à gérer les zones occultées, en utilisant un réseau enseignant et un réseau élève. Dans un premier temps, le réseau enseignant est entraîné en utilisant l'erreur photométrique ignorant les zones occultées, comme précé-

demment. Ensuite, lors de l'entraînement du réseau élève, des zones des images sont artificiellement masquées (ces zones sont obtenues par une décomposition en superpixels pour avoir une forme cohérente avec les structures de l'image) avant l'estimation. Dans ces zones masquées, le réseau élève est "supervisé" par l'estimation du réseau enseignant qui, lui, prend en entrée les images initiales, sans masquage, et a donc l'information nécessaire pour estimer le mouvement. Dans les zones non masquées, l'entraînement du réseau élève utilise l'erreur photométrique. Cette auto-supervision permet à SelFlow d'apprendre à estimer le mouvement dans des régions occultées, il paraît vraisemblable que SelFlow apprenne à opérer une sorte d'*inpainting* spatial en utilisant l'information du voisinage pour combler le manque d'information photométrique dans ces zones. En utilisant également la transformée census et l'estimation aller-retour du flot, comme proposé par MEISTER et collab. [2018], une variante multiframe de l'architecture PWC-Net à la manière de JANAI et collab. [2018], et en utilisant comme données d'entraînement des séquences du film d'animation duquel est issu le *dataset* MPI Sintel, SelFlow (LIU et collab. [2019]) est la première méthode non supervisée rivalisant avec les méthodes supervisées sur MPI Sintel, et établit le nouvel état de l'art au moment de sa publication.

2.3.5 Occultations

Plusieurs articles, comme IRR-PWC dont nous avons déjà parlé, étudient la question de la détection des occultations dans les méthodes de flot optique par apprentissage profond. Dans cette section, nous présentons les architectures des modules dédiés à cette détection dans la littérature.

La méthode ContinualFlow (NEORAL et collab. [2018]), basée sur PWC-Net, propose l'ajout d'un module d'estimation des occultations à chaque étage du décodeur du réseau et utilise le résultat en entrée de l'estimateur de flot, ces modules apparaissent en rose dans la partie inférieure du schéma de ContinualFlow représenté en figure 2.15. Ce module est un CNN qui prend en entrée les scores de corrélation, la carte d'occultation de l'échelle précédente et le flot de l'échelle précédente, pour réaliser une classification à deux classes (occulté et non occulté). Ensuite l'estimateur de flot prend en entrée la carte d'occultations estimée, en plus de ses entrées habituelles (scores de corrélation, descripteurs de l'encodeur, flot de l'échelle précédente), l'idée des auteurs étant d'améliorer les performances de l'estimateur de flot grâce à la connaissance de la carte d'occultations. Leurs expériences montrent une amélioration importante des résultats de flot optique grâce à l'ajout de ce module d'occultations. ContinualFlow propose également un mécanisme multiframe, dont nous parlerons en section 2.4.2.

HUR et ROTH [2019] réalisent l'estimation des occultations, en plus du flot, en ajoutant, à chaque étage du décodeur de PWC-Net, un bloc convolutif dense quasi identique à celui d'estimation du flot pour en faire un estimateur d'occultations, figure 2.14. Ce bloc diffère dans sa couche de sortie qui ne présente qu'un canal au lieu de 2 et qui ajoute l'application d'une fonction *sigmoid*, donnant des valeurs entre 0 (pixel visible) et 1 (pixel occulté). Sa sortie est donc une carte donnant la probabilité d'occultation en chaque pixel. Les entrées des blocs de flot et d'occultation sont les mêmes à l'exception du flot de l'échelle précédente qui est remplacé par la carte d'occultations de l'échelle précédente. Leur architecture est donc formée d'un encodeur commun suivi de deux décodeurs séparés, d'architectures quasi identiques, fonctionnant en parallèle. L'un des décodeurs est dédié à l'estimation d'occultations, l'autre à l'estimation de flot. Les deux prennent les mêmes descripteurs en entrée, issus de l'encodeur commun et de la couche de corrélation. Leurs expériences montrent une amélioration de l'estimation du flot grâce au simple

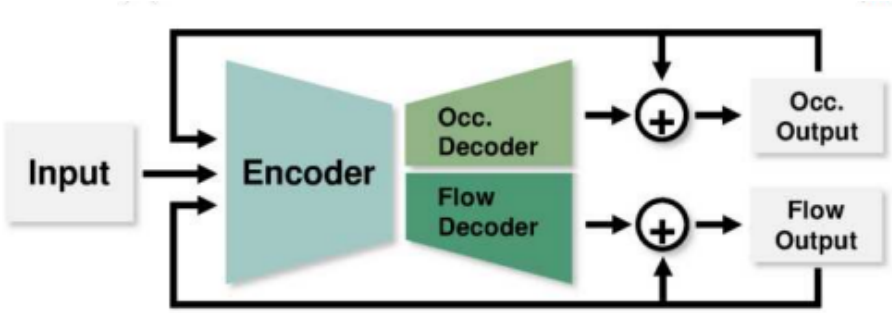

FIGURE 2.14 – Schéma de principe de l'estimation conjointe du flot et des occultations proposée par HUR et ROTH [2019] pour leur architecture IRR-PWC. Source : HUR et ROTH [2019].

ajout du bloc d'estimation des occultations, sans utiliser la carte d'occultations estimée pour l'estimation du flot. Il est possible que l'ajout d'une tâche supervisée supplémentaire, et qui est fortement liée à la première, conduise à l'apprentissage de meilleurs descripteurs — dans l'encodeur qui est commun — pour les deux tâches. Ils proposent, aussi, une version de leur méthode dans laquelle la carte d'occultation est rééchantillonnée, par un réseau, à pleine résolution (plutôt que de rester au quart de la résolution image, comme pour le flot), cela améliore encore les résultats de flot.

ILG et collab. [2018] ont exploré la question de l'estimation des occultations en plus du flot, en utilisant FlowNet. Leurs travaux comparent 3 instances de FlowNetC entraînées différemment : la première apprend à estimer le flot seulement, la deuxième les occultations seulement, la troisième apprend à estimer les deux conjointement. Dans les trois cas le réseau ne prend que la paire d'images en entrée. Les performances sont comparées pour le flot entre les méthodes 1 et 3, et pour les occultations entre les méthodes 2 et 3. Les résultats sont sensiblement les mêmes, ces expériences indiquent donc que l'ajout de l'estimation de l'une des tâches n'améliore par l'estimation de l'autre. Ils en déduisent que le réseau de flot apprend implicitement à gérer les occultations (et vice-versa). Ces conclusions sont opposées à plusieurs travaux récents (NEORAL et collab. [2018] et HUR et ROTH [2019]) qui indiquent, au contraire, que la détection des occultations profite à l'estimation du flot optique. Nous serons amenés à revenir sur cette question dans le chapitre 5 en section 5.2.

Les trois méthodes citées ci-dessus sont entraînées de manière supervisée pour le flot et les occultations. Or la vérité terrain des occultations est disponible dans le jeu de données FlyingThings3D mais pas dans FlyingChairs. NEORAL et collab. [2018] ont donc réalisé le pré-entraînement sur FlyingChairs sans estimation d'occultations, et ont entraîné la totalité du modèle, avec les occultations, sur FlyingThings3D. Quant à eux, HUR et ROTH [2019] et ILG et collab. [2018] ont chacun proposé une nouvelle version de FlyingChairs, en y ajoutant la vérité terrain des cartes d'occultations.

Comme nous l'avons vu un peu plus tôt, les méthodes non supervisées ont recours à une estimation d'occultations pour les éliminer du calcul de l'erreur photométrique dans la fonction de coût. Certaines méthodes (LIU et collab. [2019]; MEISTER et collab. [2018]) estiment le flot dans les deux sens et détectent les occultations en vérifiant la cohérence entre les flots aller et retour. La méthode de JANAI et collab. [2018] apprend de manière non supervisée à estimer les occultations : leur architecture, comme celle de HUR et ROTH [2019], dispose d'un décodeur supplémentaire, dédié à l'estimation des occultations.

Enfin, très récemment, ZHAO et collab. [2020] proposent MaskFlownet, une méthode supervisée pour l'estimation de flot optique, mais utilisant un mécanisme auto-supervisé

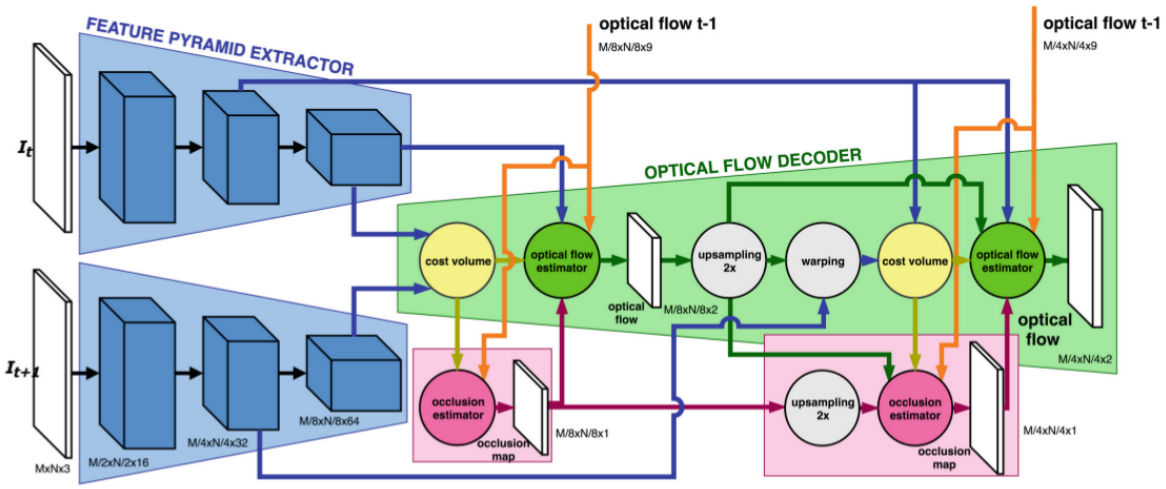

FIGURE 2.15 – Schéma de principe de ContinualFlow. Source : NEORAL et collab. [2018].

pour apprendre à estimer une carte d'occultations. Cette dernière est utilisée pour filtrer le résultat du rééchantillonnage (*warping*) des descripteurs de l'image cible avant calcul de la corrélation, afin d'enlever les structures dédoublées dans les zones occultées ("ghosting effect").

2.4 Estimation de mouvement multiframe

Pour la plupart, les algorithmes d'estimation du flot optique n'utilisent que deux images pour réaliser leur estimation. Quelques autres méthodes tirent profit d'une continuité temporelle qui, dans de nombreux cas pratiques en vidéo, s'étend sur plus de deux images successives.

Certaines méthodes se placent dans un contexte multiframe en considérant une séquence de plus de deux images pour améliorer la qualité de l'estimation. Ainsi, WANG et collab. [2008] obtiennent une estimation moins bruitée, et une réduction du nombre d'appariements ambigus, en augmentant le nombre d'images. Une réduction du bruit de mesure est également constatée par YEGAVIAN et collab. [2016], dans le contexte de la mesure de champs de vitesses par imagerie en mécanique des fluides. VOLZ et collab. [2011] montrent une amélioration de l'estimation du mouvement dans les régions peu texturées, grâce à une régularisation temporelle. Enfin, KENNEDY et TAYLOR [2015] et REN et collab. [2019] améliorent les résultats grâce à l'utilisation d'instants supplémentaires, en particulier dans les zones d'occultations. Le mouvement d'un objet visible à l'instant t mais occulté à l'instant $t+1$ ne peut être correctement estimé en considérant l'information photométrique entre t et $t+1$, cependant en considérant d'autres instants, où ce même objet a de grandes chances d'être visible (à $t-1$ par exemple), on peut prédire son mouvement entre t et $t+1$ en modélisant l'évolution temporelle.

Nous détaillons un peu plus dans les deux sections qui suivent certaines des approches utilisées en multiframe.

2.4.1 Dans les méthodes classiques

Certains travaux associent à une séquence vidéo un ensemble de trajectoires collectées dans une matrice $\boldsymbol{W} = \begin{bmatrix} \mathrm{U} \\ \mathrm{V} \end{bmatrix}$ de taille $2N \times P$ contenant les coordonnées images de P points dans N images (à N instants successifs). Lorsque la caméra suit un modèle de projection orthographique, TOMASI et KANADE [1992] montrent que cette matrice est de rang 3. Une première idée est donc d'utiliser des contraintes de rang afin de réduire la dimension de l'espace des trajectoires possibles.

Dans le cas d'un mouvement rigide, IRANI [2002] montre que la matrice $\boldsymbol{W}$ est de rang faible et applique la contrainte de rang sur une extension de l'équation de conservation de l'intensité au cadre multiframe. D'après IRANI [2002] cela permet de résoudre l'indétermination du problème de l'estimation du flot optique grâce à la dimension temporelle, en trouvant une base des trajectoires et en estimant non plus le flot mais ses coefficients sur cette base. Cette idée est transposée au cas non rigide par GARG et collab. [2010] qui suppose que, même dans ce cas, le rang est suffisamment faible pour contraindre l'estimation. La méthode proposée commence par extraire des points facilement reconnaissables (coins, zones texturées) et estimer leur trajectoire tout au long de la séquence. Une base de modèles temporels est estimée par analyse en composantes principales sur ces trajectoires. Puis le flot aux différents instants est estimé par une méthode de type Horn-Schunk en remplaçant le flot par son expression sur cette base. Il existe, enfin, une extension de FOLKI au cas multiframe, *Lucas-Kanade Fluid Trajectories* (LKFT) , proposée par YEGAVIAN et collab. [2016], développée dans le cadre de la PIV en mécanique des fluides. Cet algorithme suppose que les deux composantes des trajectoires sont des fonctions polynomiales du temps. Les coefficients sur cette base polynomiale sont estimés par une méthode de type FOLKI direct. L'article propose des comparaisons entre LKFT (multiframe) et FOLKI (biframe). LKFT apporte un gain en termes de réduction du bruit. Dans les méthodes proposées par IRANI [2002], GARG et collab. [2010] et YEGAVIAN et collab. [2016], les inconnues sont donc les coefficients de la décomposition du mouvement sur une base de modèles temporels au lieu du déplacement à chaque instant. On réduit ainsi le nombre d'inconnues à calculer.

La méthode de VOLZ et collab. [2011] tient compte de l'information temporelle en ajoutant un terme de régularisation temporelle à l'énergie à minimiser de l'équation (2.24). Ce terme impose au flot de varier peu entre instants temporellement proches.

Enfin, KENNEDY et TAYLOR [2015] supposent le flot constant sur un horizon temporel de 4 instants ($t-1$, t, $t+1$, $t+2$) et fusionnent les estimations suivantes : l'opposé du flot estimé de t vers $t-1$, le flot estimé entre t et $t+1$ et le flot entre t et $t+2$ divisé par 2.

Dans le chapitre 3 nous généralisons l'algorithme LKFT au cas d'une décomposition du mouvement sur une base de modèles temporels arbitraires, et utilisons une analyse en composantes principales pour apprendre une telle base à partir des données étudiées.

2.4.2 Avec les réseaux de neurones convolutifs

Parmi les méthodes d'estimation de flot optique par apprentissage profond, un faible nombre propose de considérer plus de deux images consécutives pour améliorer l'estimation. MAURER et BRUHN [2018] et REN et collab. [2019] proposent des méthodes exploitant l'information temporelle en fusionnant plusieurs estimations. Dans la méthode de MAURER et BRUHN [2018], un réseau est entraîné pour prédire le flot de t à $t+1$ à partir du flot de t vers $t-1$ et d'une carte d'occultations. L'estimation finale est issue de la fusion de deux flots : le flot estimé directement entre t et $t+1$ (par un algorithme biframe) et le

flot prédit à partir du flot estimé entre t et $t-1$. Un peu différemment, la méthode de REN et collab. [2019] utilise le flot de $t-1$ vers t, qui est rééchantillonné dans la géométrie de l'instant t et fusionné au flot estimé entre t et $t+1$.

Les méthodes non supervisées de JANAI et collab. [2018] et LIU et collab. [2019] considèrent 3 images et estiment conjointement les flots de t vers $t-1$ et de t vers $t+1$.

ContinualFlow (NEORAL et collab. [2018]) propose d'ajouter des connections temporelles dans le réseau PWC-Net : à chaque étage du décodeur, l'estimateur de flot prend une entrée supplémentaire qui est le flot estimé à l'instant précédent à l'échelle correspondante, voir figure 2.15. Le flot de l'instant précédent est, au préalable, rééchantillonné dans la géométrie de l'instant courant. Cette méthode a la particularité de présenter un fonctionnement temporellement récurrent.

Toutes les méthodes présentées ci-dessus, proposant une estimation multiframe par apprentissage profond, ne tiennent compte que de 3 instants. Seule la méthode ContinualFlow fonctionne différemment, pour estimer le flot entre t et $t+1$ elle considère les deux images à t et $t+1$ et le flot de l'instant précédent, entre $t-1$ et t. Autrement dit elle transmet de manière récurrente le flot estimé d'un instant au suivant. S'il est possible que ce fonctionnement entraîne une amélioration sur plus de 3 instants par améliorations successives, l'information retenue du passé se limite tout de même à l'instant précédent. Les chapitres 3 à 5 consistent en de nouvelles propositions pour faire de l'estimation de mouvement multiframe. Nous abordons notamment la question de l'utilisation d'un horizon croissant d'images (suivant les algorithmes, $N = 4$ à $N = 7$ images sont impliquées dans le processus d'estimation) et celle de la sortie produite par le réseau : trajectoires ou succession de flots instantanés.

2.5 Références

BAILER, C., B. TAETZ et D. STRICKER. 2015, «Flow Fields : Dense correspondence fields for highly accurate large displacement optical flow estimation», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 4015–4023. 26

BAKER, S. et I. MATTHEWS. 2004, «Lucas-Kanade 20 years on : A unifying framework», *International journal of computer vision*, vol. 56, nº 3, p. 221–255. 23

BAKER, S., D. SCHARSTEIN, J. LEWIS, S. ROTH, M. J. BLACK et R. SZELISKI. 2007, «A database and evaluation methodology for optical flow», *International Conference on Computer Vision*. 18, 26

BAR-HAIM, A. et L. WOLF. 2020, «ScopeFlow : Dynamic scene scoping for optical flow», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 7998–8007. 34, 37

BENGIO, Y., J. LOURADOUR, R. COLLOBERT et J. WESTON. 2009, «Curriculum learning», dans *Proceedings of the 26th annual international conference on machine learning*, p. 41–48. 34

BLACK, M. J. et P. ANANDAN. 1993, «A framework for the robust estimation of optical flow», dans *1993 (4th) International Conference on Computer Vision*, IEEE, p. 231–236. 14

BOUGUET, J.-Y. 2001, «Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm», *Intel Corporation*, vol. 5, nº 1-10, p. 4. 23

BROX, T., C. BREGLER et J. MALIK. 2009, «Large displacement optical flow», dans *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, p. 41–48. 26, 27, 32

BROX, T., A. BRUHN, N. PAPENBERG et J. WEICKERT. 2004, «High accuracy optical flow estimation based on a theory for warping», *Computer Vision-ECCV 2004*, p. 25–36. 14, 25, 27, 33

BUTLER, D. J., J. WULFF, G. B. STANLEY et M. J. BLACK. 2012, «A naturalistic open source movie for optical flow evaluation», dans *European Conference on Computer Vision*, Springer, p. 611–625. 15, 18, 26

CHAMPAGNAT, F., A. PLYER, G. LE BESNERAIS, B. LECLAIRE, S. DAVOUST et Y. LE SANT. 2011, «Fast and accurate PIV computation using highly parallel iterative correlation maximization», *Experiments in fluids*, vol. 50, nº 4, p. 1169. 27

CORDTS, M., M. OMRAN, S. RAMOS, T. REHFELD, M. ENZWEILER, R. BENENSON, U. FRANKE, S. ROTH et B. SCHIELE. 2016, «The cityscapes dataset for semantic urban scene understanding», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 3213–3223. 38

DOSOVITSKIY, A., P. FISCHER, E. ILG, P. HAUSSER, C. HAZIRBAS, V. GOLKOV, P. VAN DER SMAGT, D. CREMERS et T. BROX. 2015, «FlowNet : Learning optical flow with convolutional networks», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 2758–2766. 21, 28, 29, 30, 31, 32, 34

EIGEN, D., C. PUHRSCH et R. FERGUS. 2014, «Depth map prediction from a single image using a multi-scale deep network», dans *Advances in neural information processing systems*, p. 2366–2374. 28

GARG, R., L. PIZARRO, D. RUECKERT et L. AGAPITO. 2010, «Dense multi-frame optic flow for non-rigid objects using subspace constraints», dans *Asian Conference on Computer Vision*, Springer, p. 460–473. 42

GEIGER, A., P. LENZ et R. URTASUN. 2012, «Are we ready for autonomous driving? the KITTI vision benchmark suite», dans *Conference on Computer Vision and Pattern Recognition (CVPR)*. 18

GOODFELLOW, I., Y. BENGIO et A. COURVILLE. 2016, *Deep Learning*, MIT Press. http://www.deeplearningbook.org. 28

HORN, B. K. et B. G. SCHUNCK. 1981, «Determining optical flow», *Artificial intelligence*, vol. 17, nº 1-3, p. 185–203. 16, 24, 25

HUANG, G., Z. LIU, K. Q. WEINBERGER et L. VAN DER MAATEN. 2017, «Densely connected convolutional networks», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, p. 3. 35

HUI, T.-W., X. TANG et C. C. LOY. 2018, «LiteFlowNet : A lightweight convolutional neural network for optical flow estimation», dans *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 34

HUR, J. et S. ROTH. 2019, «Iterative residual refinement for joint optical flow and occlusion estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5754–5763. ix, 34, 36, 37, 39, 40

ILG, E., N. MAYER, T. SAIKIA, M. KEUPER, A. DOSOVITSKIY et T. BROX. 2017, «FlowNet 2.0 : Evolution of optical flow estimation with deep networks», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 6. 21, 33, 34, 36

ILG, E., T. SAIKIA, M. KEUPER et T. BROX. 2018, «Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation», dans *European Conference on Computer Vision*, Springer, p. 626–643. 40

IRANI, M. 2002, «Multi-frame correspondence estimation using subspace constraints», *International Journal of Computer Vision*, vol. 48, nº 3, p. 173–194. 42

JANAI, J., F. GUNEY, A. RANJAN, M. BLACK et A. GEIGER. 2018, «Unsupervised learning of multi-frame optical flow with occlusions», dans *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 690–706. 38, 39, 40, 43

JASON, J. Y., A. W. HARLEY et K. G. DERPANIS. 2016, «Back to basics : Unsupervised learning of optical flow via brightness constancy and motion smoothness», dans *European Conference on Computer Vision*, Springer, p. 3–10. 38

KENNEDY, R. et C. J. TAYLOR. 2015, «Optical flow with geometric occlusion estimation and fusion of multiple frames», dans *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, p. 364–377. 41, 42

KRIZHEVSKY, A., I. SUTSKEVER et G. E. HINTON. 2012, «Imagenet classification with deep convolutional neural networks», dans *Advances in neural information processing systems*, p. 1097–1105. 28

LE BESNERAIS, G. et F. CHAMPAGNAT. 2005, «Dense optical flow by iterative local window registration», dans *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, IEEE, p. I–137. 23

LIU, P., M. R. LYU, I. KING et J. XU. 2019, «SelFlow : Self-supervised learning of optical flow», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 34, 38, 39, 40, 43

LUCAS, B. D., T. KANADE et collab.. 1981, «An iterative image registration technique with an application to stereo vision», . 16, 22

MAURER, D. et A. BRUHN. 2018, «ProFlow : Learning to predict optical flow», dans *Proceedings of the British Machine Vision Conference*. 42

MAYER, N., E. ILG, P. HAUSSER, P. FISCHER, D. CREMERS, A. DOSOVITSKIY et T. BROX. 2016, «A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4040–4048. 32, 33

MEISTER, S., J. HUR et S. ROTH. 2018, «UnFlow : Unsupervised learning of optical flow with a bidirectional census loss», dans *Thirty-Second AAAI Conference on Artificial Intelligence*. 38, 39, 40

MENZE, M. et A. GEIGER. 2015, «Object scene flow for autonomous vehicles», dans *Conference on Computer Vision and Pattern Recognition*. 13, 15, 18, 26

NEORAL, M., J. ŠOCHMAN et J. MATAS. 2018, «Continual occlusion and optical flow estimation», dans *Asian Conference on Computer Vision*, Springer, p. 159–174. 34, 39, 40, 41, 43

PLYER, A., G. LE BESNERAIS et F. CHAMPAGNAT. 2016, «Massively parallel Lucas Kanade optical flow for real-time video processing applications», *Journal of Real-Time Image Processing*, vol. 11, n° 4, p. 713–730. 27

RANJAN, A. et M. J. BLACK. 2017, «Optical flow estimation using a spatial pyramid network», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. 34

REN, Z., O. GALLO, D. SUN, M.-H. YANG, E. SUDDERTH et J. KAUTZ. 2019, «A fusion approach for multi-frame optical flow estimation», dans *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, p. 2077–2086. 34, 41, 42, 43

REN, Z., J. YAN, B. NI, B. LIU, X. YANG et H. ZHA. 2017, «Unsupervised deep learning for optical flow estimation», dans *Thirty-First AAAI Conference on Artificial Intelligence*. 38

REVAUD, J., P. WEINZAEPFEL, Z. HARCHAOUI et C. SCHMID. 2015, «Epicflow : Edge-preserving interpolation of correspondences for optical flow», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 1164–1172. 26, 27, 32

REVAUD, J., P. WEINZAEPFEL, Z. HARCHAOUI et C. SCHMID. 2016, «Deepmatching : Hierarchical deformable dense matching», *International Journal of Computer Vision*, vol. 120, nº 3, p. 300–323. 26

RONNEBERGER, O., P. FISCHER et T. BROX. 2015, «U-Net : Convolutional networks for biomedical image segmentation», dans *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, p. 234–241. 31

ROS, G., L. SELLART, J. MATERZYNSKA, D. VAZQUEZ et A. M. LOPEZ. 2016, «The synthia dataset : A large collection of synthetic images for semantic segmentation of urban scenes», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 3234–3243. 38

SHI, J. et collab.. 1994, «Good features to track», dans *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, IEEE, p. 593–600. 23

SIMONYAN, K. et A. ZISSERMAN. 2014, «Very deep convolutional networks for large-scale image recognition», *arXiv preprint arXiv :1409.1556*. 29

SUN, D., S. ROTH et M. J. BLACK. 2014, «A quantitative analysis of current practices in optical flow estimation and the principles behind them», *International Journal of Computer Vision*, vol. 106, nº 2, p. 115–137. 25, 36

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018, «PWC-Net : CNNs for optical flow using pyramid, warping, and cost volume», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 8934–8943. 13, 21, 34

TAO, X., H. GAO, R. LIAO, J. WANG et J. JIA. 2017, «Detail-revealing deep video super-resolution», dans *The IEEE International Conference on Computer Vision (ICCV)*. 28

TOMASI, C. et T. KANADE. 1992, «Shape and motion from image streams under orthography : a factorization method», *International Journal of Computer Vision*, vol. 9, nº 2, p. 137–154. 42

UMMENHOFER, B., H. ZHOU, J. UHRIG, N. MAYER, E. ILG, A. DOSOVITSKIY et T. BROX. 2017, «Demon : Depth and motion network for learning monocular stereo», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5038–5047. 28

VOLZ, S., A. BRUHN, L. VALGAERTS et H. ZIMMER. 2011, «Modeling temporal coherence for optical flow», dans *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, p. 1116–1123. 41, 42

WANG, C.-M., K.-C. FAN, C.-T. WANG et collab.. 2008, «Estimating optical flow by integrating multi-frame information», *Journal of Information Science and Engineering*, vol. 24, nº 6, p. 1719–1731. 41

WEINZAEPFEL, P., J. REVAUD, Z. HARCHAOUI et C. SCHMID. 2013, «Deepflow : Large displacement optical flow with deep matching», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 1385–1392. 26, 27

YEGAVIAN, R., B. LECLAIRE, F. CHAMPAGNAT, C. ILLOUL et G. LOSFELD. 2016, «Lucas–kanade fluid trajectories for time-resolved PIV», *Measurement science and Technology*, vol. 27, nº 8, p. 084 004. 41, 42

YU, F., V. KOLTUN et T. A. FUNKHOUSER. 2017, «Dilated residual networks», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 3. 36

ZABIH, R. et J. WOODFILL. 1994, «Non-parametric local transforms for computing visual correspondence», dans *European conference on computer vision*, Springer, p. 151–158. 38

ZACH, C., T. POCK et H. BISCHOF. 2007, «A duality based approach for realtime TV-L1 optical flow», *Pattern Recognition*, p. 214–223. 27

ŽBONTAR, J. et Y. LECUN. 2016, «Stereo matching by training a convolutional neural network to compare image patches», *The journal of machine learning research*, vol. 17, nº 1, p. 2287–2318. 28

ZHAO, S., Y. SHENG, Y. DONG, E. I. CHANG, Y. XU et collab.. 2020, «MaskFlownet : Asymmetric feature matching with learnable occlusion mask», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 6278–6287. 40

Chapitre 3

Algorithme de Lucas-Kanade multiframe à dépendance temporelle apprise

Sommaire

Ce chapitre porte sur des travaux issus d'une collaboration avec Frédéric Champagnat. Nous tenons à remercier Philippe Cornic, Olivier Marquet et Benjamin Leclaire pour leur aide précieuse quant à l'obtention des données simulées utilisées dans ce chapitre pour l'évaluation des algorithmes.

3.1 Introduction

Nous avons vu qu'il existait deux grandes familles de méthodes basées modèle pour l'estimation du mouvement. Bien que les méthodes locales soient en général moins robustes que les méthodes globales en cas de manque de texture ou au niveau des discontinuités spatiales du champs de mouvement, elles présentent l'intérêt d'être plus rapides, plus intuitives à paramétrer (taille de la fenêtre), et plus faciles à mettre en œuvre et à caractériser (il est plus facile de comprendre le comportement d'une méthode par corrélation de fenêtre qu'une estimation dépendant d'une optimisation globale d'une fonctionnelle parfois complexe). Ces arguments font des méthodes locales la solution retenue dans les applications de mesure par imagerie, pour lesquelles il est fréquent de texturer artificiellement l'objet de la mesure, ou d'inséminer le milieu d'intérêt. En mécanique des fluides, la vélocimétrie par images de particules ou *Particle Image Velocimetry* (PIV) est une technique permettant de mesurer le champ de vitesses d'un écoulement. Des particules, nommées traceurs, sont introduites dans le fluide, et éclairées par des impulsions laser. En éclairant une zone de faible épaisseur selon une des trois dimensions, on se restreint à un plan du fluide, comme illustré sur le schéma de la figure 1.4 dans le chapitre 1. La zone éclairée par la nappe laser est photographiée à deux instants rapprochés, aux moments des flashs lasers. On utilise alors, en général, une méthode par corrélation de fenêtre pour estimer le mouvement des particules entre les deux images, et en déduire le champ de vitesses. Une branche de la PIV, appelée "time-resolved" consiste à acquérir des séquences d'images à très haute cadence, permettant la mesure de phénomènes d'échelles temporelles très courtes. Cependant, il est courant que de telles séquences présentent un mauvais rapport signal à bruit, d'où l'intérêt des méthodes multiframe profitant de la redondance de l'information pour réduire le bruit de mesure par régularisation temporelle. Les méthodes existantes (JEON et collab. [2014]; LYNCH et SCARANO [2013]; YEGAVIAN et collab. [2016]) font l'hypothèse d'une dépendance temporelle polynomiale des trajectoires. Nous allons voir que, dans certains cas, cette hypothèse peut être inexacte en pratique et conduire à des biais de mesure. Nous reprendrons ensuite le formalisme de l'algorithme *Lucas-Kanade Fluid Trajectories* (LKFT) (YEGAVIAN et collab. [2016]) afin de montrer qu'il est généralisable à l'utilisation d'une décomposition non polynomiale. La généralité de ce nouveau formalisme permet de décomposer la dépendance temporelle du mouvement sur une base de modèles temporels arbitraires, et qui peut donc être apprise à partir des données d'intérêt. Dans le contexte de la vision par ordinateur, il a été montré que la dépendance temporelle du mouvement pouvait être caractérisée par un très faible nombre de vecteurs de base, qu'on peut extraire par analyse en composantes principales ou *Principal Component Analysis* (PCA) (GARG et collab. [2010]; IRANI [2002]; TOMASI et KANADE [1992]). Cela nous conduira à proposer un nouvel algorithme multiframe profitant d'un a priori temporel non polynomial, appris à partir de la séquence d'images dont on cherche à extraire le mouvement.

3.2 Dépendance temporelle polynomiale en vélocimétrie par images de particules

3.2.1 Présentation de l'algorithme LKFT

L'algorithme local FOLKI, présenté en 2.2.1, a été adapté et appliqué au contexte de la PIV par CHAMPAGNAT et collab. [2011]. Une extension multiframe, supposant une dé-

pendance temporelle polynomiale, en a ensuite été proposée sous le nom de LKFT par YEGAVIAN et collab. [2016]. Ces derniers travaux ont montré, dans le contexte de la PIV, que la prise en compte d'une séquence de $N > 2$ images permettait une réduction de la variance de l'estimateur.

Dans l'algorithme à deux images FOLKI, $\boldsymbol{u}$ représente un champ de déplacements, et est donc indexé par deux dimensions spatiales $\boldsymbol{k}$. Pour l'algorithme LKFT nous considérons des champs denses de **trajectoires**, u et v ont donc deux dimensions spatiales et une dimension temporelle. Ainsi $\boldsymbol{u}(k,l,t_n)$ est le vecteur déplacement 2D, entre l'instant de référence t_{ref} et l'instant t_n, de l'élément fluide visible au pixel $\boldsymbol{k}$ de l'image de référence. On a donc $\boldsymbol{u}(t_{ref}) = 0$, et $\boldsymbol{u}(t_n)$ est un tableau de dimensions $H \times W \times 2$ (où H et W sont respectivement la hauteur et la largeur des images) représentant le flot entre les instants t_{ref} et t_n, *ie* permettant le recalage de I_n et I_{ref} par interpolation de I_n.

LKFT reprend le critère, à minimiser, de FOLKI : la somme quadratique des différences d'intensités lumineuses dans une fenêtre $\mathcal{W}$ et ajoute une somme sur les N instants $(t_n = t_0 + n\Delta t)_{0\leq n\leq N-1}$ de la séquence considérée. Ceci en fait un critère multiframe pour chaque pixel $\boldsymbol{k} = \begin{pmatrix} k \\ l \end{pmatrix}$:

$$\sum_{n=1}^{N} \sum_{\boldsymbol{k}'} \mathcal{W}(\boldsymbol{k}-\boldsymbol{k}') \left(I_{ref}(\boldsymbol{k}') - I_n(\boldsymbol{k}' + \boldsymbol{u}(\boldsymbol{k}, t_n)) \right)^2 \tag{3.1}$$

Chacun des termes de la somme sur n correspond au critère à deux images de FOLKI, faisant intervenir une image choisie pour référence (commune pour tous les termes de la somme) et chacune des autres images de la séquence, comme le montre la figure 3.1. C'est généralement l'image centrale de la séquence qui est choisie pour référence, puisque l'amplitude maximale des déplacements considérés, pour une même séquence, sera plus faible si on prend l'instant central pour référence (et donc comme origine des déplacements) plutôt qu'un instant extrême.

L'hypothèse polynomiale est introduite en écrivant chacune des composantes (horizontale et verticale) des trajectoires comme un polynôme, de degré d, de la variable temporelle :

$$u(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(u)}_{\boldsymbol{k},i} t_n^i \tag{3.2}$$

$$v(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(v)}_{\boldsymbol{k},i} t_n^i \tag{3.3}$$

On utilisera également la notation suivante, plus compacte

$$\boldsymbol{u}(\boldsymbol{k}, t_n) = \mathrm{T}_n \boldsymbol{\theta}_{\boldsymbol{k}} \tag{3.4}$$

avec $\boldsymbol{\theta}_{\boldsymbol{k}}$ de dimension $2d \times 1$, les coefficients

$$\boldsymbol{\theta}_{\boldsymbol{k}} = [\theta^{(u)}_{\boldsymbol{k},1}, \dots \theta^{(u)}_{\boldsymbol{k},d}, \theta^{(v)}_{\boldsymbol{k},1}, \dots, \theta^{(v)}_{\boldsymbol{k},d}]^\mathsf{T} \tag{3.5}$$

et T_n la matrice de dimension $2 \times 2d$ représentant la base polynomiale de degré d :

$$\mathrm{T}_n = \begin{bmatrix} \boldsymbol{E}_n^\mathsf{T} & 0 \\ 0 & \boldsymbol{E}_n^\mathsf{T} \end{bmatrix} \text{ avec } \boldsymbol{E}_n = [(t_n - t_{ref}) \cdots (t_n - t_{ref})^d]^\mathsf{T}. \tag{3.6}$$

L'expression polynomiale (3.4) est introduite dans l'expression du critère (3.1), et on cherche le jeu de coefficients $\boldsymbol{\theta}_{\boldsymbol{k}}$ qui minimise ce critère. On cherche donc à estimer d

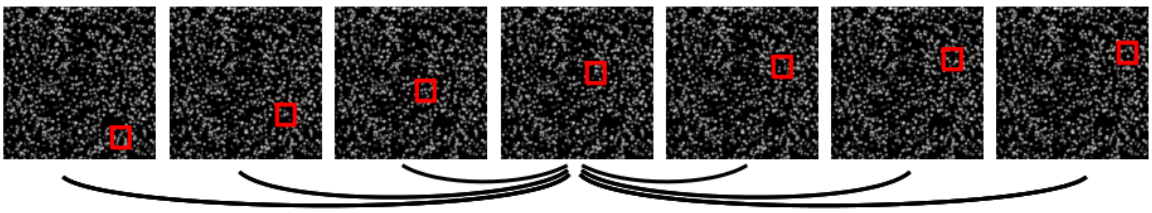

FIGURE 3.1 – Estimation multiframe avec LKFT. Ici on choisit l'image centrale comme image de référence. L'estimation de la trajectoire de l'élément fluide en chaque pixel de l'image de référence consiste à minimiser un critère décrivant la somme des carrés des différences d'intensité, avec chacune des autres images, dans une fenêtre (représentée en rouge) autour de ce pixel.

coefficients par pixel, pour représenter les trajectoires sur N instants. Il s'agit là d'une réduction de dimension, puisqu'en pratique $d \approx 2,3$ et N est de l'ordre de la dizaine. Cette réduction de dimension a deux vertus. D'une part, elle diminue les coûts calculatoires puisqu'on a moins de valeurs à estimer et à stocker. D'autre part elle contraint l'estimation en réduisant l'espace dans lequel on recherche des solutions, et revient ici à imposer une régularisation temporelle.

3.2.2 Réduction du bruit de mesure grâce à la redondance temporelle avec LKFT

Les travaux de YEGAVIAN et collab. [2016] montrent que l'utilisation d'instants supplémentaires, avec une modélisation polynomiale de l'évolution temporelle, permet d'améliorer les résultats, plus précisément de limiter la variance de l'estimateur, dans le cas de données bruitées. C'est la conséquence positive de la contrainte de rang faible (réduction de dimension). Nous avons reproduit avec notre implémentation d'une forme inverse de l'algorithme LKFT les résultats présentés par YEGAVIAN et collab. [2016] sur les données du troisième *PIV Challenge* (*case B*), STANISLAS et collab. [2008]. Il s'agit d'une séquence de PIV issue d'une simulation DNS (*Direct Numerical Simulation*), composée de 120 images, et dont le rapport signal à bruit (RSB) diminue par palier toutes les 20 images. En réalité, la même séquence de 20 images est répétée 6 fois, avec des RSB différents. On dispose d'une vérité terrain (VT) éparse (sur une grille régulière de pas supérieur à celui de la grille des pixels) pour l'instant central de chaque palier de RSB, c'est-à-dire entre les instants 10 et 11, 30 et 31, ..., 110 et 111. On réalise l'estimation du mouvement en ces instants afin de comparer les performances dans les différentes conditions de RSB, voir figure 3.2. On représente l'évolution de l'erreur RMS sur le champ horizontal $u(t_{ref+1})$ entre t_{ref} et $t_{ref+1} = t_{ref} + \Delta t$,

$$RMS(u) = \sqrt{\frac{1}{K}\sum_{k}(u[k] - u_{vt}[k])^2} \tag{3.7}$$

pour les K pixels pour lesquels on dispose de la VT [1], en fonction de la position de l'image de référence (qui prend les valeurs discrètes $10,30,50,70,90,110$), donc en fonction du RSB, pour différents algorithmes :

— FOLKI (entre l'instant de référence et le suivant) ;

— LKFT tenant compte de $N = 5$ instants avec un modèle polynomial de degré $d = 2$;

— LKFT tenant compte de $N = 15$ instants avec un modèle polynomial de degré $d = 3$.

1. Afin d'ignorer les effets de bord, on calcule l'erreur en prenant une marge au bord, correspondant à 10 % des dimensions de l'image.

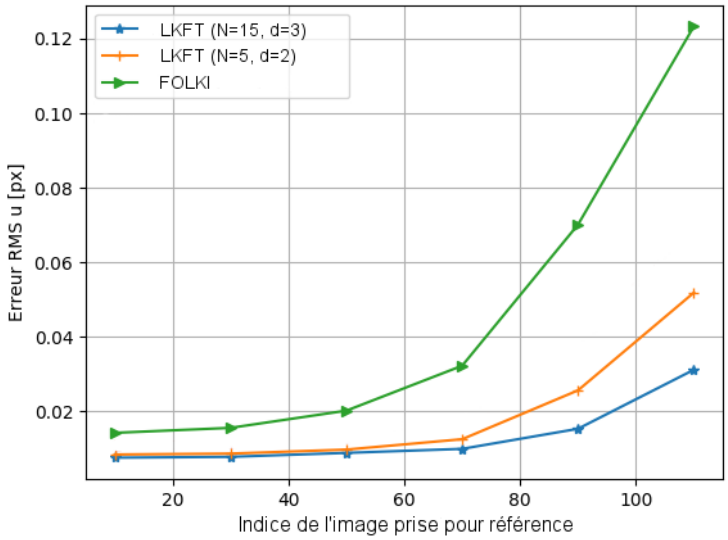

FIGURE 3.2 – Erreur RMS sur le déplacement horizontal $u(t_{ref+1})$ estimé entre les instants t_{ref} et $t_{ref+1} = t_{ref} + \Delta t$ sur la séquence du PIV Challenge 2005. Le rapport signal à bruit des images de la séquence diminue par palier toutes les 20 images. Toutes les méthodes utilisent une fenêtre uniforme carrée de 33 pixels de côté.

Pour LKFT, les N instants sont choisis autour de l'instant de référence, *ie* on prend l'image de référence et $\frac{N-1}{2}$ images de chaque côté. Généralement, pour augmenter significativement le nombre d'images et continuer à observer une réduction de l'erreur, il faut également augmenter le degré du polynôme, sans quoi l'erreur finirait par réaugmenter. Le résultat serait trop régularisé, par un modèle trop simple pour modéliser la complexité de longues trajectoires. On remarque sur la figure 3.2 que l'augmentation du nombre d'images considérées permet de réduire l'erreur RMS sur u, en particulier lorsque le RSB est faible (derniers paliers, autour des images 90 et 110).

La figure 3.3 montre visuellement la réduction de la variance de l'estimateur lorsqu'on augmente le nombre d'images. Précisons qu'ici, et dans tout ce chapitre, l'augmentation du nombre d'images ne correspond pas à un changement de la cadence vidéo, mais à étendre la fenêtre temporelle par l'ajout d'instants supplémentaires de plus en plus éloignés de l'image de référence.

NB1 : Notons que les résultats de LKFT ne sont pas obtenus avec l'algorithme décrit par YEGAVIAN et collab. [2016] mais en utilisant une forme inverse décrite plus loin dans ce chapitre, en 3.3.1.
NB2 : la notation N de YEGAVIAN et collab. [2016] représente le nombre d'images de chaque côté de l'image de référence, tandis que notre notation N représente le nombre total d'images de la séquence considérée par l'algorithme. On a donc $N = 2N_{Yegavian} + 1$.

3.2.3 Mise en évidence d'un biais dû à l'hypothèse polynomiale

Dans l'exemple précédent le champ de mouvement est plutôt lisse spatialement et temporellement. Il s'agit d'un cas où l'hypothèse polynomiale donne de très bons résultats. Nous allons présenter une séquence mettant en scène un écoulement plus complexe et montrer que l'hypothèse polynomiale, alors mise en défaut, peut causer l'apparition

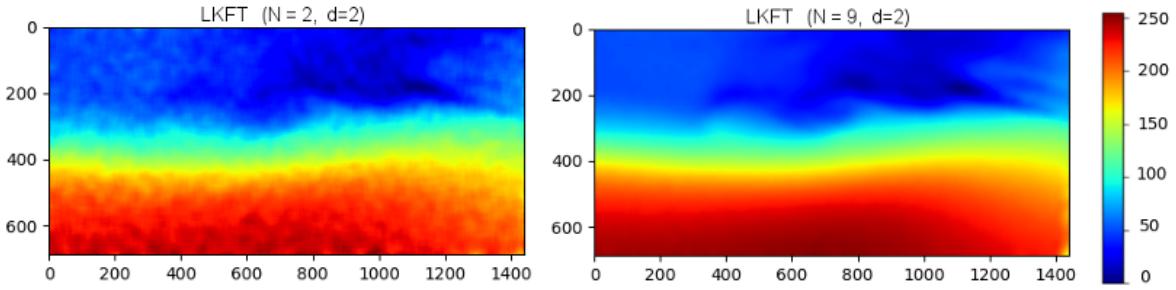

FIGURE 3.3 – Pour la séquence du troisième *PIV Challenge* (*case B*), STANISLAS et collab. [2008], $ref = 90$, comparaison des champs de déplacements horizontaux $u(t_{ref+1})$, entre les instants t_{ref} et t_{ref+1}, estimés avec FOLKI (biframe) et LKFT (multiframe). La redondance de l'information permet de profiter d'une régularisation temporelle et d'obtenir un résultat moins bruité. Les deux méthodes utilisent une fenêtre uniforme carrée de 33 pixels de côté.

d'un biais de mesure localisé.

Présentation de la séquence de l'aile battante

Le but de cette section est de proposer une séquence vidéo d'un écoulement à 2 dimensions et qui présente un niveau suffisant de complexité pour mettre en défaut l'hypothèse polynomiale. Nous proposons de synthétiser une séquence d'images de particules suivant l'écoulement fluide du sillage d'une aile battante, en partant de champs de vitesses simulés par des collègues du département DAAA (Aérodynamique, aéroélasticité, acoustique) de l'ONERA (JALLAS et collab. [2017]). Ces champs de vitesses sont obtenus par simulation DNS de l'écoulement, à nombre de Reynolds égal à 250.

On dispose de champs de vitesses instantanées (discrets) pour 30 instants successifs, et on souhaite synthétiser des images représentant des particules qui suivent ces champs de vitesses. Pour cela on part d'un ensemencement initial, dans le plan d'une nappe laser virtuelle, en tirant aléatoirement les positions des particules selon une distribution uniforme dans un plan couvrant largement le champ d'un modèle de caméra défini par ailleurs, et de sorte à avoir un nombre moyen de particules par pixel de 0.1. On définit ensuite une échelle de temps et on calcule les positions des particules à chaque instant. La position d'une particule à l'instant $t+1$ est obtenue par intégration (Runge-Kutta) de la vitesse évaluée à l'instant t et à la position de cette particule à t. Cette position à l'instant précédent n'étant pas forcément sur la grille discrète où la vitesse est connue, on réalise une interpolation bilinéaire des champs de vitesses. Les images, aux différents instants, sont synthétisées en projetant les particules dans l'espace image d'un modèle de caméra, suivant un modèle sténopé. Un paramètre important de cette étape est la taille de la PSF (*Point Spread Function* pour "Fonction d'étalement du point") des particules, cette PSF est simulée par une gaussienne dont on règle l'écart-type. Pour la séquence utilisée, les images des particules sont des gaussiennes de 0.4 pixels d'écart-type.

Pour s'approcher, un peu, d'images réalistes, et pouvoir évaluer la robustesse au bruit, les intensités des particules sont tirées aléatoirement, et un bruit additif gaussien est ajouté aux images. Quand rien n'est précisé, le niveau de bruit obtenu dans la séquence étudiée est de 4% de l'intensité maximale.

Les paramètres de la simulation sont choisis de sorte que la séquence soit bien échantillonnée temporellement, avec des déplacements inter-image de l'ordre du pixel, afin d'être représentative d'une expérience de PIV résolue en temps où l'on aurait intérêt à utiliser un algorithme multiframe.

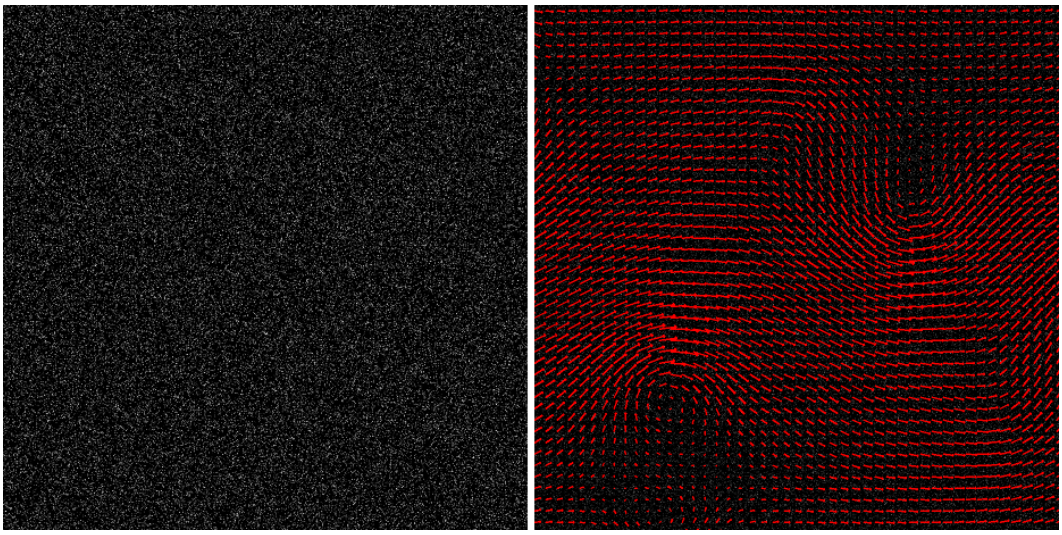

FIGURE 3.4 – Séquence de l'aile battante : image et champ de déplacements de l'instant central.

Les images étant obtenues par simulation, nous pouvons générer une VT des champs de déplacements, dans l'espace image. Pour cela, on rétro-projette les positions des pixels de la caméra dans le plan de la nappe laser virtuelle, où l'on interpole et intègre le champ de vitesses. On obtient ainsi, dans le repère de la nappe laser, les positions à l'instant $t+1$ de particules fictives dont les images seraient situées aux positions de chaque pixel de l'image à l'instant t. En projetant ces nouvelles positions dans le plan de la caméra, et par différence avec les positions initiales (celles de la grille des pixels), on obtient le champ de déplacements VT entre t et $t+1$.

La figure 3.4 montre l'image centrale de la séquence obtenue et la VT du champ de déplacements, la figure 3.5 détaille les composantes horizontale (direction de l'écoulement) et verticale du déplacement. L'instant central, t_{ref}, est celui qui sera pris pour référence dans la suite. Dans la figure 3.5, la région encadrée dans la partie gauche est un exemple de région où les gradients spatiaux du champ de mouvement sont faibles, tandis que la région encadrée de droite présente des gradients spatiaux élevés.

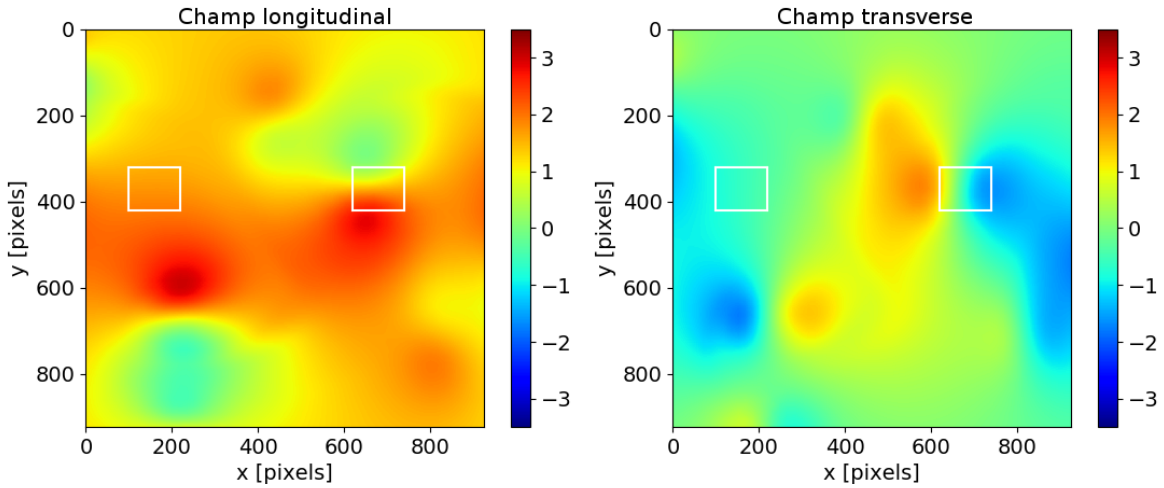

FIGURE 3.5 – Vérité terrain du champ de déplacements entre t_{ref} et t_{ref+1}, pour deux instants consécutifs de la séquence de l'aile battante. Composante longitudinale (dans la direction de l'écoulement) à gauche et composante transverse à droite.

Résultats de FOLKI et LKFT sur la séquence de l'aile battante

On peut comparer, sur la figure 3.6, les estimations par la méthode biframe FOLKI et par la méthode multiframe LKFT, du champ de déplacements vertical (transverse à l'écoulement) à l'instant central, *ie* entre t_{ref} et $t_{ref+1} = t_{ref} + \Delta t$, de la séquence de l'aile battante. LKFT donne en chaque pixel une trajectoire sur N instants, mais ici nous ne considérons que le déplacement $\boldsymbol{u}(t_{ref+1})$ entre l'instant de référence et l'instant suivant, quantité comparable au déplacement mesuré par la méthode biframe appliquée entre ces deux instants. Plus précisément, la figure 3.6 montre les cartes d'erreur obtenues (distance euclidienne par rapport à la VT). Un premier constat : l'estimation multiframe donne un résultat moins bruité que l'estimation biframe, ce qui est cohérent avec les expériences précédentes. On remarque cependant, dans l'estimation de LKFT, des erreurs très localisées qui correspondent aux zones du champ où les gradients spatiaux du champ de vitesses sont élevés, voir figure 3.5.

Afin de montrer que ces erreurs sont dues à une mise en défaut localisée de l'hypothèse polynomiale, nous allons mesurer une erreur de modèle en calculant le résidu de la projection polynomiale.

Analyse de l'erreur de modèle

Dans cette section on met de côté les images pour ne s'intéresser qu'aux trajectoires simulées et à leur approximation polynomiale. On cherche à caractériser l'erreur de modèle en considérant le résidu de la décomposition polynomiale.

Grâce à la VT du déplacement, on construit, pour chaque pixel $\boldsymbol{k}$ de l'image choisie pour référence, une trajectoire sur N instants. Pour estimer la meilleure représentation polynomiale de degré d de cette trajectoire, on cherche les coefficients $\boldsymbol{\theta}_{\boldsymbol{k}}$ qui minimisent l'écart à la trajectoire au sens des moindres carrés :

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{k}}^{(u)} = \arg\min_{\boldsymbol{\theta}} \sum_n (\mathrm{T}_n\boldsymbol{\theta} - u_{VT}(\boldsymbol{k}, t_n))^2 \tag{3.8}$$

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{k}}^{(v)} = \arg\min_{\boldsymbol{\theta}} \sum_n (\mathrm{T}_n\boldsymbol{\theta} - v_{VT}(\boldsymbol{k}, t_n))^2 \tag{3.9}$$

On en déduit le résidu de la décomposition polynomiale :

$$\text{résidu}(\boldsymbol{k}, t_n) = \sqrt{\left(u_{VT}(\boldsymbol{k}, t_n) - \sum_{i=1}^{d} \hat{\theta}_{\boldsymbol{k},i}^{(u)} t_n^i\right)^2} \tag{3.10}$$

Les résultats présentés précédemment, en figure 3.6, comparent les estimations de FOLKI et LKFT entre les instants t_{ref} et t_{ref+1}, nous allons donc illustrer, sur la figure 3.7, la répartition spatiale du résidu à l'instant t_{ref+1}. On remarque, en comparant les figures 3.6 et 3.7, que les zones où LKFT présente une erreur locale élevée correspondent à des zones où l'erreur de modèle est élevée. Cette erreur de modèle donne une borne inférieur de l'erreur atteignable avec une représentation polynomiale de degré 3 des trajectoires. Nous avons mis en évidence la présence d'un biais de mesure local, aux endroits du champ où le modèle utilisé ne permet pas une bonne représentation des trajectoires.

Une solution à ce problème pourrait être de trouver un modèle temporel plus adapté, lorsque le modèle polynomial est mis en défaut. Encore faut-il pouvoir calculer l'estimateur de mouvement correspondant à ce modèle, ce que nous allons faire en généralisant le formalisme de LKFT.

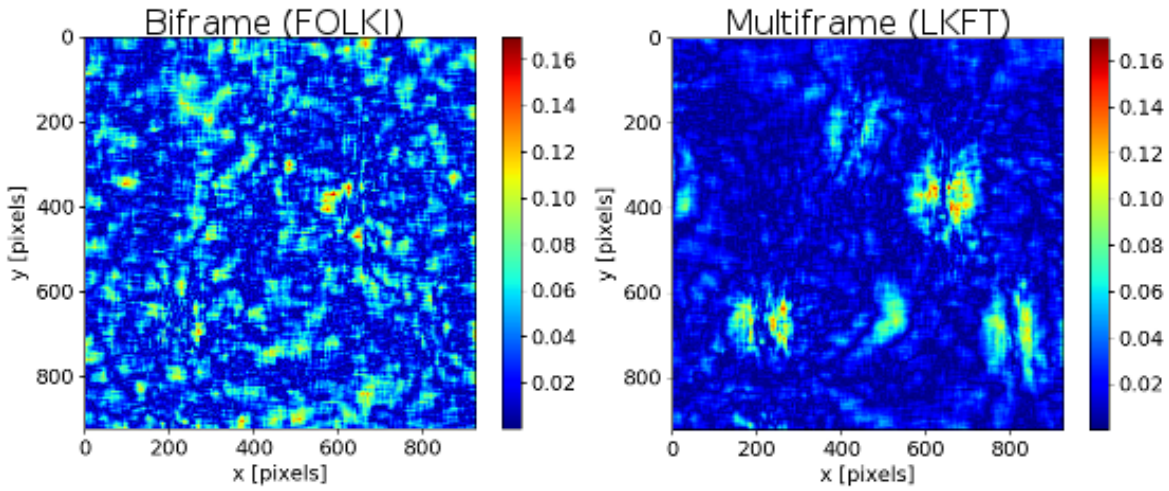

FIGURE 3.6 – Comparaison des cartes d'erreur (en pixels) de FOLKI (CHAMPAGNAT et collab. [2011]) et LKFT (YEGAVIAN et collab. [2016]) pour le champ transverse $v(t_{ref+1})$ de l'instant central de la séquence de l'aile battante. Les deux algorithmes utilisent une fenêtre uniforme carrée de 33 pixels de côté. LKFT est paramétré avec une fenêtre temporelle de $N = 8$ instants et un degré de polynôme $d = 3$.

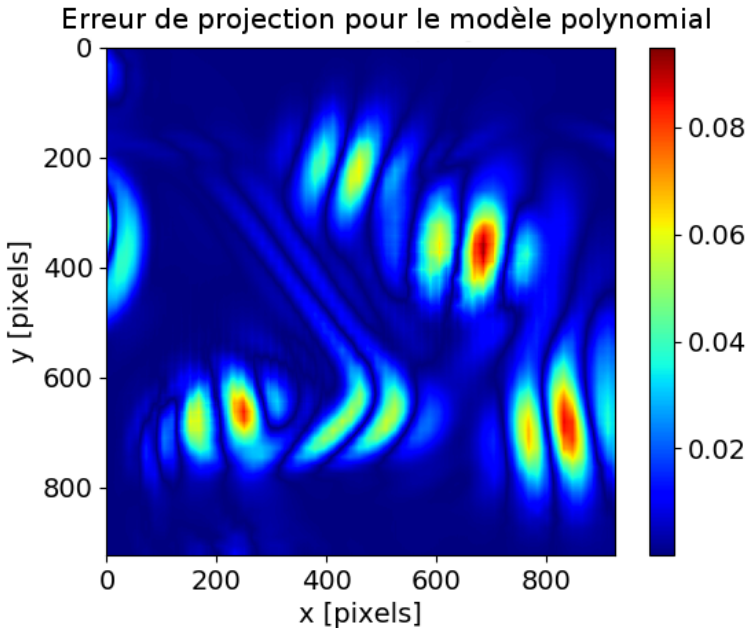

FIGURE 3.7 – Résidu de la décomposition polynomiale (erreur de modèle), pour le champ transverse de la séquence de l'aile battante, entre les instants t_{ref} et t_{ref+1}.

3.3 Généralisation du formalisme de l'algorithme LKFT

Nous allons ici développer les équations du formalisme de l'algorithme LKFT, afin de montrer qu'il est généralisable au cas d'une base non polynomiale. Avant cela, nous allons voir que l'utilisation de la forme inverse permet une formulation plus simple et une implémentation plus efficace de l'algorithme.

3.3.1 La forme inverse

Nous avons vu en 2.2.1 que, dans le cas biframe, la forme inverse permettait de réduire significativement le nombre de calculs de gradients et d'interpolations par rapport à la forme directe. Rappelons que pour l'algorithme de Lucas-Kanade, le critère à minimiser de la forme **directe** linéarisée s'écrit :

$$\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\Big(I_{ref}(\boldsymbol{k})-I_n(\boldsymbol{k}+\boldsymbol{u}_0)-\nabla I_n(\boldsymbol{k}+\boldsymbol{u}_0)^{\mathsf{T}}\boldsymbol{\delta u}\Big)^2 \tag{3.11}$$

et celui de la forme **inverse** linéarisée :

$$\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\Big(I_{ref}(\boldsymbol{k})-I_n(\boldsymbol{k}+\boldsymbol{u}_0)-\nabla I_{ref}(\boldsymbol{k})^{\mathsf{T}}\boldsymbol{\delta u}\Big)^2 \tag{3.12}$$

Dans le cas multiframe, cette expression est sommée sur l'indice n. On remarque alors que la forme directe implique de calculer une image de gradients différente pour chaque terme de cette somme sur n, soit autant de calculs supplémentaires qu'il y a d'images dans l'horizon temporel choisi. Dans la forme inverse, au contraire, chaque terme fait intervenir les gradients de l'image de référence, une quantité commune pour tous les instants de la séquence, que l'on peut calculer une fois pour toutes. Le gain de la forme inverse par rapport à la forme directe est donc encore plus significatif dans le cas multiframe.

Nous allons donc écrire la forme inverse du formalisme de LKFT, présenté par Yegavian et collab. [2016] dans sa forme directe. LKFT étant une extension multiframe de l'algorithme FOLKI, nous allons en fait étendre au cas multiframe la forme inverse du critère de FOLKI proposée par Champagnat et collab. [2011]. Par rapport à l'expression de l'équation (3.12), donnée ci-dessus pour la forme inverse de Lucas-Kanade, cela revient à remplacer $\boldsymbol{\delta u}$ par $\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}^{(0)}(\boldsymbol{k}')$ et à sommer sur les instants de la séquence :

$$\sum_{n=1}^{N}\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\Big(\varepsilon(\boldsymbol{k}',n)-\nabla I_{ref}(\boldsymbol{k}')^{\mathsf{T}}\cdot\boldsymbol{u}(\boldsymbol{k},t_n)\Big)^2 \tag{3.13}$$

avec

$$\varepsilon(\boldsymbol{k}',n)=I_{ref}(\boldsymbol{k}')-\tilde{I}_n(\boldsymbol{k}'+\boldsymbol{u}^{(0)}(\boldsymbol{k}',t_n))+\nabla I_{ref}(\boldsymbol{k}')^{\mathsf{T}}\cdot\boldsymbol{u}^{(0)}(\boldsymbol{k}',t_n) \tag{3.14}$$

En introduisant la décomposition polynomiale des équations (3.4), (3.5), (3.6), on obtient le critère suivant à minimiser

$$\begin{aligned} J_{\boldsymbol{k}}(\boldsymbol{\theta}) &= \sum_{n=1}^{N}\sum_{\boldsymbol{k}'\in\mathscr{W}(\boldsymbol{k})}\Big(\varepsilon(\boldsymbol{k}',n)-\nabla I_{ref}(\boldsymbol{k}')^{\mathsf{T}}\mathrm{T}_n\boldsymbol{\theta}\Big)^2 &(3.15)\\ &= \boldsymbol{\theta}^{\mathsf{T}}\mathrm{A}_{\boldsymbol{k}}\boldsymbol{\theta}-2\boldsymbol{b}_{\boldsymbol{k}}^{\mathsf{T}}\boldsymbol{\theta}+\text{constante}(\boldsymbol{\theta}) &(3.16)\end{aligned}$$

La solution permettant d'annuler la dérivée de ce critère, est donnée en chaque pixel par le jeu de coefficients

$$\boldsymbol{\theta}_{\boldsymbol{k}}=\mathrm{A}_{\boldsymbol{k}}^{-1}\boldsymbol{b}_{\boldsymbol{k}}, \tag{3.17}$$

avec $\mathrm{A}_{\boldsymbol{k}}$ la matrice $2d \times 2d$, d étant le degré du modèle polynomial utilisé

$$\mathrm{A}_{\boldsymbol{k}} = \sum_{n=1}^{N} \sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \mathrm{T}_n^{\mathsf{T}} \nabla I_{ref}(\boldsymbol{k}') \nabla I_{ref}(\boldsymbol{k}')^{\mathsf{T}} \mathrm{T}_n \tag{3.18}$$

$$= \sum_{n=1}^{N} \mathrm{T}_n^{\mathsf{T}} \left(\sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \nabla I_{ref}(\boldsymbol{k}') \nabla I_{ref}(\boldsymbol{k}')^{\mathsf{T}} \right) \mathrm{T}_n \tag{3.19}$$

et $\boldsymbol{b}_{\boldsymbol{k}}$ est le vecteur de dimension $2d$ donné par

$$\boldsymbol{b}_{\boldsymbol{k}} = \sum_{n=1}^{N} \sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \mathrm{T}_n^{\mathsf{T}} \nabla I_{ref}(\boldsymbol{k}') \varepsilon(\boldsymbol{k}', n) \tag{3.20}$$

$$= \sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \left(\sum_{n=1}^{N} \mathrm{T}_n \varepsilon(\boldsymbol{k}', n) \right)^{\mathsf{T}} \nabla I_{ref}(\boldsymbol{k}') \tag{3.21}$$

la forme factorisée de $\mathrm{A}_{\boldsymbol{k}}$ est possible car T_n ne dépend que du temps, et pas de la position dans l'image. Ceci vient de la séparabilité temps-espace de la décomposition (3.4). La forme factorisée de $\boldsymbol{b}_{\boldsymbol{k}}$ nécessite cette même séparabilité, mais également que $\nabla I_{ref}(\boldsymbol{k}')$ soit indépendant du temps, ce qui est vrai dans notre cas grâce à l'utilisation de la forme inverse, mais ne le serait pas pour la forme directe. La première expression de $\boldsymbol{b}_{\boldsymbol{k}}$, non factorisée, nécessite le calcul de $2N$ convolutions ; la deuxième expression, factorisée, en nécessite $2d$. Dans la pratique le nombre N d'images de la séquence est plus élevé que le degré d du polynôme, on a tout intérêt à implémenter cette deuxième expression. Cela représente un avantage supplémentaire de la forme inverse.

3.3.2 Utilisation d'une base quelconque

Les formules énoncées ci-dessus permettent l'estimation des coefficients de la décomposition du mouvement de (3.4) :

$$\boldsymbol{u}(\boldsymbol{k}, t_n) = \mathrm{T}_n \boldsymbol{\theta}_{\boldsymbol{k}} \tag{3.4}$$

c'est-à-dire selon un modèle temporel que nous avons, dans le cas de LKFT, choisi polynomial, et qui est représenté par la matrice T_n. Ce même formalisme permet-il l'utilisation d'un autre modèle temporel, simplement en modifiant l'expression de T_n ? Sous quelles conditions ?

L'équation (3.6) donne l'expression de T_n pour le cas polynomial de LKFT, dans lequel on utilise le même modèle pour les composantes horizontale et verticale du mouvement. Dans un souci de généralité, on en donne ici une expression dans laquelle des modèles différents peuvent être utilisés sur les deux axes :

$$\mathrm{T}_n = \begin{bmatrix} \boldsymbol{E}_n^{\mathsf{T}} & 0 \\ 0 & \boldsymbol{H}_n^{\mathsf{T}} \end{bmatrix} \text{ avec } \boldsymbol{E}_n = [\boldsymbol{e}_1[n] \cdots \boldsymbol{e}_d[n]]^{\mathsf{T}} \text{ et } \boldsymbol{H}_n = [\boldsymbol{h}_1[n] \cdots \boldsymbol{h}_d[n]]^{\mathsf{T}}. \tag{3.22}$$

Les $\boldsymbol{e}_i$ et $\boldsymbol{h}_i$ sont des vecteurs de $\mathbb{R}^N$, dont la donnée simultanée représente une trajectoire (discrétisée) sur N instants. La valeur à l'instant n de la coordonnée horizontale (resp. verticale) est donnée par la n-ième composante de $\boldsymbol{e}_i$ (resp. $\boldsymbol{h}_i$).

Estimer les coefficients $\boldsymbol{\theta}$ correspondant à une telle expression de T_n revient à décomposer l'évolution temporelle de la composante horizontale u (resp. verticale v) du mouvement comme une combinaison linéaire de d vecteurs de $\mathbb{R}^N$ $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ (resp. $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d$).

Ce même jeu de vecteurs est utilisé pour tous les pixels, et un jeu de d coefficients est estimé par pixel.

$$u(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(u)}_{\boldsymbol{k},i} \boldsymbol{e}_i[n] \tag{3.23}$$

$$v(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(v)}_{\boldsymbol{k},i} \boldsymbol{h}_i[n] \tag{3.24}$$

Le calcul du jeu de coefficients, dont la solution est donnée par (3.17), suppose la matrice $\mathrm{A}_{\boldsymbol{k}}$ inversible. Nous allons étudier dans quels cas cette hypothèse est vérifiée, et en particulier ce que cela implique sur la matrice T_n. Dans le cas biframe, l'inversibilité de $\mathrm{A}_{\boldsymbol{k}}$ est liée à la texture de l'image : l'estimation d'un mouvement dans une direction donnée nécessite que le gradient image soit non nul dans la direction correspondante. On peut se demander si l'ajout d'instants supplémentaires relâche cette contrainte dans certains cas. Dans le cas où le gradient image est nul dans une direction seulement (matrice de covariance des gradients de rang 1), l'estimation biframe sera ambigüe dans cette direction, mais si la trajectoire est courbée on peut supposer qu'une estimation multiframe lève l'ambiguïté. Nous analysons cette question dans l'annexe B.

3.3.3 Inversibilité de la matrice $\mathrm{A}_{\boldsymbol{k}}$

Nous montrons, dans l'annexe B que l'inversibilité de la matrice $\mathrm{A}_{\boldsymbol{k}}$ est assurée dès lors que

1. la matrice de covariance des gradients $\left(\sum_{\boldsymbol{k}' \in F(\boldsymbol{k})} \nabla I_{ref}(\boldsymbol{k}')(\nabla I_{ref})^{\mathsf{T}}(\boldsymbol{k}')\right)$ est inversible.
2. les jeux de vecteurs $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ et $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d$ définissent chacun une base d'un sous-espace de dimension d dans $\mathbb{R}^N$.

Dans le contexte de la métrologie, les données étant artificiellement texturées, on peut raisonnablement supposer que la matrice de covariance des gradients est inversible.

Il est donc possible de remplacer la décomposition polynomiale par une combinaison linéaire de d modèles temporels quelconques $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ dès lors que ceux-ci forment une base d'un sous-espace de dimension d dans $\mathbb{R}^N$. Il s'agit donc toujours de contraindre l'estimation du mouvement par une réduction de dimension en modélisant l'évolution temporelle, mais la dépendance temporelle peut être non polynomiale. Ce nouveau formalisme inclut celui de LKFT puisqu'il est possible d'utiliser une base polynomiale : $e_i[n] = h_i[n] = t_n^i$. La possibilité d'utiliser n'importe quelle base rend possible l'utilisation d'un modèle temporel appris à partir des données.

Nous appelons FOLKI-MF, pour FOLKI MultiFrame, cette version généralisée du formalisme de LKFT. Il s'agit d'une extension multiframe de l'algorithme FOLKI permettant de contraindre l'estimation en modélisant les trajectoires comme des combinaisons linéaires de vecteurs d'une base dont le choix est libre. Le paramètre d, qui correspondait au degré du polynôme pour LKFT, correspond maintenant au nombre de vecteurs qui composent la base.

Simplification dans le cas d'une base commune pour les deux axes

Dans la méthode présentée, l'estimation des coefficients en chaque pixel est basée sur l'inversion de la matrice $\mathrm{A}_{\boldsymbol{k}}$, de dimension $2d \times 2d$. Cette opération est coûteuse, et l'est d'autant plus qu'elle doit être réalisée à chaque itération. Nous allons voir qu'il est possible de réduire les coûts calculatoires en factorisant cette matrice en une matrice 2×2

dépendant de $\boldsymbol{k}$, et une matrice $d \times d$ indépendante de $\boldsymbol{k}$ dont on peut calculer l'inverse une seule fois pour toutes les itérations et tous les pixels. Ainsi, à chaque itération et en chaque pixel, il ne reste qu'à inverser la matrice 2×2, ce qui est beaucoup moins coûteux.

Nous commençons par proposer une factorisation de $\mathrm{A}_{\boldsymbol{k}}$ dans le cas où la même base est utilisée pour l'axe horizontal et l'axe vertical. Dans ce cas on a

$$\mathrm{T}_n = \begin{bmatrix} \boldsymbol{E}_n^\mathsf{T} & 0 \\ 0 & \boldsymbol{E}_n^\mathsf{T} \end{bmatrix} \text{ avec } \boldsymbol{E}_n = [\boldsymbol{e}_1[n] \cdots \boldsymbol{e}_d[n]]^\mathsf{T}. \tag{3.25}$$

La matrice

$$\mathrm{A}_{\boldsymbol{k}} = \sum_{n=1}^{N} \mathrm{T}_n^\mathsf{T} \left(\sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \nabla I_{ref}(\boldsymbol{k}') \nabla I_{ref}(\boldsymbol{k}')^\mathsf{T} \right) \mathrm{T}_n \tag{3.19}$$

peut être factorisée, grâce à la séparabilité temps-espace, comme

$$\mathrm{A}_{\boldsymbol{k}} = \begin{bmatrix} \alpha_{\boldsymbol{k}} \mathbf{T} & \beta_{\boldsymbol{k}} \mathbf{T} \\ \beta_{\boldsymbol{k}} \mathbf{T} & \gamma_{\boldsymbol{k}} \mathbf{T} \end{bmatrix}, \tag{3.26}$$

en notant :

$$\begin{bmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{bmatrix} = \sum_{\boldsymbol{k}'} \nabla I_{ref}(\boldsymbol{k}') \nabla I_{ref}(\boldsymbol{k}')^\mathsf{T} \tag{3.27}$$

et $\mathbf{T}$ la matrice $d \times d$:

$$\mathbf{T} = \sum_n \boldsymbol{E}_n^\mathsf{T} \boldsymbol{E}_n \tag{3.28}$$

On obtient ainsi l'expression suivante pour l'inverse :

$$\mathrm{A}_{\boldsymbol{k}}^{-1} = \frac{1}{\alpha_{\boldsymbol{k}} \gamma_{\boldsymbol{k}} - \beta_{\boldsymbol{k}}^2} \begin{bmatrix} \gamma_{\boldsymbol{k}} \mathbf{T}^{-1} & -\beta_{\boldsymbol{k}} \mathbf{T}^{-1} \\ -\beta_{\boldsymbol{k}} \mathbf{T}^{-1} & \alpha_{\boldsymbol{k}} \mathbf{T}^{-1} \end{bmatrix}, \tag{3.29}$$

La partie la plus coûteuse de l'inversion de $\mathrm{A}_{\boldsymbol{k}}$ est le calcul de $\mathbf{T}^{-1}$, qui peut être réalisé une fois pour tous les pixels et itérations. À chaque itération, et pour chaque pixel, il ne restera qu'à calculer $\alpha_{\boldsymbol{k}}$, $\beta_{\boldsymbol{k}}$ and $\gamma_{\boldsymbol{k}}$ et en déduire $\mathrm{A}_{\boldsymbol{k}}^{-1}$.

Le cas de deux bases différentes

La simplification précédente n'est pas valable dans le cas de deux bases différentes, on peut cependant proposer une autre factorisation, nécessitant plus de calculs mais permettant tout de même de pré-calculer les quantités les plus coûteuses en amont des itérations.

On reprend les équations (3.19) et (3.21) en considérant maintenant deux bases différentes :

$$\mathrm{T}_n = \begin{bmatrix} \boldsymbol{E}_n^\mathsf{T} & 0 \\ 0 & \boldsymbol{H}_n^\mathsf{T} \end{bmatrix} \text{ avec } \boldsymbol{E}_n = [\boldsymbol{e}_1[n] \cdots \boldsymbol{e}_d[n]]^\mathsf{T} \text{ et } \boldsymbol{H}_n = [\boldsymbol{h}_1[n] \cdots \boldsymbol{h}_d[n]]^\mathsf{T}. \tag{3.30}$$

L'enjeu est donc de décomposer l'inversion de

$$\mathrm{A}_{\boldsymbol{k}} = \sum_{n=1..N} \mathrm{T}_n^\mathsf{T} \begin{bmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{bmatrix} \mathrm{T}_n = \begin{bmatrix} \alpha_{\boldsymbol{k}} \mathrm{S} & \beta_{\boldsymbol{k}} \mathrm{R} \\ \beta_{\boldsymbol{k}} \mathrm{R}^\mathsf{T} & \gamma_{\boldsymbol{k}} \mathrm{T} \end{bmatrix} \tag{3.31}$$

avec

$$\mathrm{S} = \sum_{n=1..N} \boldsymbol{E}_n \boldsymbol{E}_n^\mathsf{T} \tag{3.32}$$

$$\mathrm{T} = \sum_{n=1..N} \boldsymbol{H}_n \boldsymbol{H}_n^\mathsf{T} \tag{3.33}$$

$$\mathrm{R} = \sum_{n=1..N} \boldsymbol{E}_n \boldsymbol{H}_n^\mathsf{T} \tag{3.34}$$

Posons

$$\mathrm{A}_{\boldsymbol{k}}^{-1} = \begin{bmatrix} \mathrm{M}_{11} & \mathrm{M}_{12} \\ \mathrm{M}_{21} & \mathrm{M}_{22} \end{bmatrix} = \begin{bmatrix} \alpha_{\boldsymbol{k}}\mathrm{S} & \beta_{\boldsymbol{k}}\mathrm{R} \\ \beta_{\boldsymbol{k}}\mathrm{R}^{\mathsf{T}} & \gamma_{\boldsymbol{k}}\mathrm{T} \end{bmatrix}^{-1} \tag{3.35}$$

D'après le lemme d'inversion des matrices partitionnées, en posant $\delta_{\boldsymbol{k}} = \frac{\beta_{\boldsymbol{k}}^2}{\alpha_{\boldsymbol{k}}\gamma_{\boldsymbol{k}}}$:

$$M_{11} = \frac{1}{\alpha_{\boldsymbol{k}}}\left(\mathrm{S} - \delta_{\boldsymbol{k}}\mathrm{R}\mathrm{T}^{-1}\mathrm{R}^{\mathsf{T}}\right)^{-1} \tag{3.36}$$

$$M_{22} = \frac{1}{\gamma_{\boldsymbol{k}}}\left(\mathrm{T} - \delta_{\boldsymbol{k}}\mathrm{R}^{\mathsf{T}}\mathrm{S}^{-1}\mathrm{R}\right)^{-1} \tag{3.37}$$

$$M_{12} = -\frac{\beta_{\boldsymbol{k}}}{\alpha_{\boldsymbol{k}}}\mathrm{S}^{-1}\mathrm{R}M_{22} \tag{3.38}$$

$$M_{21} = -\frac{\beta_{\boldsymbol{k}}}{\gamma_{\boldsymbol{k}}}\mathrm{T}^{-1}\mathrm{R}^{\mathsf{T}}M_{11} \tag{3.39}$$

Écrivons la factorisation de Cholesky pour S (symétrique définie positive) :

$$\mathrm{S} = \mathrm{S}^{\frac{1}{2}}\mathrm{S}^{\frac{\mathrm{t}}{2}} \tag{3.40}$$

On peut alors exprimer M_{11} de la manière suivante :

$$M_{11} = \frac{1}{\alpha_{\boldsymbol{k}}}\mathrm{S}^{\frac{-\mathrm{t}}{2}}\left(\mathbb{1} - \delta_{\boldsymbol{k}}\mathrm{Q}^{(u)}\right)^{-1}\mathrm{S}^{\frac{-1}{2}} \tag{3.41}$$

avec

$$\mathrm{Q}^{(u)} = \mathrm{S}^{\frac{-1}{2}}\mathrm{R}\mathrm{T}^{-1}\mathrm{R}^{\mathsf{T}}\mathrm{S}^{\frac{-\mathrm{t}}{2}} \tag{3.42}$$

T étant symétrique, $\mathrm{Q}^{(u)}$ l'est également. Or une matrice symétrique à valeurs réelles est diagonalisable en base orthonormale, on peut donc écrire

$$\mathrm{Q}^{(u)} = \mathrm{O}_u\Delta_{\mathrm{Q}}^{(u)}\mathrm{O}_u^{\mathsf{T}} \tag{3.43}$$

et

$$\boxed{M_{11} = \frac{1}{\alpha_{\boldsymbol{k}}}\mathrm{S}^{\frac{-\mathrm{t}}{2}}\mathrm{O}_u\left(\mathbb{1} - \delta_{\boldsymbol{k}}\Delta_{\mathrm{Q}}^{(u)}\right)^{-1}\mathrm{O}_u^{\mathsf{T}}\mathrm{S}^{\frac{-1}{2}}} \tag{3.44}$$

enfin en notant $\mathrm{q}_i^{(u)}$ les valeurs propres de $\mathrm{Q}^{(u)}$:

$$\left(\mathbb{1} - \delta_{\boldsymbol{k}}\Delta_{\mathrm{Q}}^{(u)}\right)^{-1} = \begin{bmatrix} \frac{1}{1-\delta_{\boldsymbol{k}}\mathrm{q}_1^{(u)}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{1-\delta_{\boldsymbol{k}}\mathrm{q}_d^{(u)}} \end{bmatrix} \tag{3.45}$$

De manière analogue on obtient :

$$\mathrm{Q}^{(v)} = \mathrm{T}^{\frac{-1}{2}}\mathrm{R}^{\mathsf{T}}\mathrm{S}^{-1}\mathrm{R}\mathrm{T}^{\frac{-\mathrm{t}}{2}} = \mathrm{O}_v\Delta_{\mathrm{Q}}^{(v)}\mathrm{O}_v^{\mathsf{T}} \tag{3.46}$$

$$\boxed{M_{22} = \frac{1}{\gamma_{\boldsymbol{k}}}\mathrm{T}^{\frac{-\mathrm{t}}{2}}\mathrm{O}_v\left(\mathbb{1} - \delta_{\boldsymbol{k}}\Delta_{\mathrm{Q}}^{(v)}\right)^{-1}\mathrm{O}_v^{\mathsf{T}}\mathrm{T}^{\frac{-1}{2}}} \tag{3.47}$$

Les étapes algorithmiques

Une fois pour toute la séquence (initialisation) :

— Choisir, ou obtenir par apprentissage, $\boldsymbol{E}_n$ et $\boldsymbol{H}_n$ pour tout n ;

— Calculer S, S^{-1}, T, T^{-1}, R ;

— Faire la factorisation de Cholesky pour S et T ;

— Calculer $\mathrm{Q}^{(u)}$ et $\mathrm{Q}^{(v)}$ puis les diagonaliser pour obtenir O_u, O_v, les $\mathrm{q}_i^{(u)}$ et les $\mathrm{q}_i^{(v)}$.

Puis en chaque pixel : calculer $\alpha_{\boldsymbol{k}}$, $\beta_{\boldsymbol{k}}$ et $\gamma_{\boldsymbol{k}}$.
Et à chaque itération calculer $\boldsymbol{b}_{\boldsymbol{k}}$ et en déduire les coefficients $\boldsymbol{\theta}$:

$$\boxed{\begin{aligned}\boldsymbol{\theta}_{\boldsymbol{k}}^{(u)} &= \mathrm{M}_{11}\boldsymbol{b}_{\boldsymbol{k}}^{(u)} - \frac{\beta_{\boldsymbol{k}}}{\alpha_{\boldsymbol{k}}}\mathrm{S}^{-1}\mathrm{R}\mathrm{M}_{22}\boldsymbol{b}_{\boldsymbol{k}}^{(v)} \\ \boldsymbol{\theta}_{\boldsymbol{k}}^{(v)} &= \mathrm{M}_{22}\boldsymbol{b}_{\boldsymbol{k}}^{(v)} - \frac{\beta_{\boldsymbol{k}}}{\gamma_{\boldsymbol{k}}}\mathrm{T}^{-1}\mathrm{R}^{\mathsf{T}}\mathrm{M}_{11}\boldsymbol{b}_{\boldsymbol{k}}^{(u)}\end{aligned}} \tag{3.48}$$

L'intérêt ici est de pré-calculer le plus de grandeurs possible (notamment les inversions et factorisations de matrices potentiellement grandes) une fois pour toutes pendant l'initialisation, afin d'économiser du temps de calcul pendant l'estimation en chaque pixel et chaque instant de la séquence.

3.4 Apprentissage de modèles temporels adaptés aux données

Nous avons vu que notre nouvel algorithme, FOLKI-MF, pouvait travailler avec n'importe quelle base de vecteurs représentant l'évolution temporelle. Nous allons donc proposer une méthode afin d'apprendre une telle base de modèles temporels à partir des données d'intérêt, et l'utiliser pour contraindre l'estimation multiframe. L'idée est de réduire, par rapport à la représentation polynomiale, le biais de mesure en proposant une base plus adaptée ; tout en continuant de profiter de la régularisation temporelle (permettant de limiter la variance de l'estimateur) en réduisant la dimension de l'espace de recherche (à un sous-espace de dimension d faible, par rapport au nombre d'images N).

3.4.1 La méthode d'extraction de la base

Nous proposons d'apprendre une base de modèles temporels par analyse en composantes principales, ou plus exactement, en utilisant une décomposition en valeurs singulières[2], SVD pour *Singular Value Decomposition*, et en ne gardant que les vecteurs correspondant aux d plus grandes valeurs singulières. Pour cela nous avons besoin d'un jeu d'exemples de trajectoires sur lequel effectuer cette SVD tronquée. Le but est d'obtenir des modèles temporels caractérisant au mieux les trajectoires qu'on cherche à estimer. On peut pour cela :

— extraire, directement de la séquence d'images étudiée, des trajectoires issues d'un suivi de points saillants (des points de Harris par exemple). En utilisant ces trajectoires pour construire notre base temporelle du mouvement et en appliquant ensuite FOLKI-MF avec cette base, on réalise une sorte de densification d'une estimation éparse de trajectoires. Dans ce cas FOLKI-MF permet d'utiliser l'information

2. Dans une analyse en composantes principales, il est d'usage de centrer les données en soustrayant le vecteur moyen. Dans notre cas d'extraction de modèles de trajectoires, nous avons réalisés quelques essais avec et sans soustraction du vecteur moyen et décidé, au vu des résultats, de ne pas centrer les données.

des trajectoires de points saillants, dont l'estimation est jugée fiable, pour guider l'estimation moins fiable des autres points.

— effectuer une simulation représentant les phénomènes qu'on s'attend à observer, et apprendre la base de modèles temporels à partir de trajectoires issues de cette simulation. Il est fréquent, dans le contexte de la mécanique des fluides, d'étudier des phénomènes à la fois par l'expérience et la simulation.

On obtient, que ce soit à partir d'un suivi de points ou d'une simulation, K échantillons de trajectoires. Cet ensemble de trajectoires doit être suffisamment grand pour être représentatif des phénomènes à étudier mais suffisamment petit pour limiter les temps de calcul. En pratique nous prenons un jeu de $K \approx 1000$ trajectoires. Chacune de ces trajectoires correspond à la données des déplacements horizontaux u et verticaux v en chaque instant. On définit alors les matrices

$$\mathrm{U} = \begin{pmatrix} u(1,t_1) & \dots & u(K,t_1) \\ \vdots & & \vdots \\ u(1,t_N) & \dots & u(K,t_N) \end{pmatrix} \quad \text{et} \quad \mathrm{V} = \begin{pmatrix} v(1,t_1) & \dots & v(K,t_1) \\ \vdots & & \vdots \\ v(1,t_N) & \dots & v(K,t_N) \end{pmatrix} \tag{3.49}$$

rassemblant les K échantillons de l'évolution temporelle, sur N instants, du déplacement horizontal (resp. vertical). On distingue alors deux approches pour effectuer l'analyse en composantes principales :

— apprendre une base unique, commune pour u et v ;

— apprendre deux bases distinctes, une pour u et l'autre pour v.

Première approche : Estimation d'une unique base issue de u et de v

On effectue une SVD sur la matrice [U | V], de dimension $N \times 2K$, il s'agit de la concaténation des deux matrices selon les colonnes, ce qui revient à mélanger les échantillons horizontaux et verticaux. On tronque cette SVD pour ne garder que les d premiers vecteurs, associés aux plus grandes valeurs singulières, afin d'obtenir une base ($\boldsymbol{e}_1$, $\boldsymbol{e}_2$, ...) issue indifféremment de u et v. On utilise donc la même base pour estimer u et v, mais on aura deux ensembles de coefficients différents :

$$u(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(u)}_{\boldsymbol{k},i} \boldsymbol{e}_i[n]$$

$$v(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(v)}_{\boldsymbol{k},i} \boldsymbol{e}_i[n]$$

Nous qualifierons d'isotrope l'algorithme utilisant ce procédé.

Deuxième approche : Estimation de deux bases distinctes pour u et v

On effectue séparément deux SVD, une sur U, l'autre sur V. On tronque ces SVD afin de ne garder que les vecteurs associés aux valeurs de plus hautes énergies. On obtient donc deux bases différentes pour u et v : $\boldsymbol{e}_1, \dots, \boldsymbol{e}_d$ et $\boldsymbol{h}_1, \dots, \boldsymbol{h}_d$ où les $\boldsymbol{e}_i$ et $\boldsymbol{h}_i$ sont des colonnes de dimension N. Pour l'estimation dense, on devra, là aussi, chercher deux ensembles de coefficients différents pour u et v :

$$u(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(u)}_{\boldsymbol{k},i} \boldsymbol{e}_i[n]$$

$$v(\boldsymbol{k}, t_n) = \sum_{i=1}^{d} \theta^{(v)}_{\boldsymbol{k},i} \boldsymbol{h}_i[n]$$

Nous qualifierons d'asymétrique l'algorithme utilisant ce procédé.

Un cadre d'apprentissage idéal

Dans la suite nous cherchons à montrer le potentiel de la méthode proposée sur la séquence simulée de l'aile battante. Nous utilisons alors un cadre idéal en constituant nos trajectoires d'apprentissage directement en échantillonnant la VT. Nous caractérisons ainsi notre méthode indépendamment de toute limitation liée à l'obtention des échantillons de trajectoires, comme un suivi de points imprécis, ou une mauvaise adéquation entre les phénomènes simulés et ceux qui régissent les données étudiées. Nous pouvons ainsi dresser une preuve de concept, montrant l'intérêt de cette méthode d'apprentissage par SVD tronquée associée à notre formalisme permettant l'utilisation d'une base apprise.

3.4.2 Réduction du biais de mesure

Retour sur l'erreur de modèle

Comme nous l'avons fait précédemment pour la base polynomiale, nous calculons la projection des trajectoires simulées de l'aile battante sur la base obtenue par SVD et en déduisons le résidu de cette représentation apprise. En comparant les deux champs d'erreur résiduelle représentés sur la ligne supérieure de la figure 3.8, on constate que notre méthode permet de réduire l'amplitude des erreurs de modèle locales, par rapport au modèle polynomial. Comme on pouvait s'y attendre la base apprise est mieux adaptée aux trajectoires présentes dans la séquence étudiée, et conduit à une meilleure approximation.

Notons tout de même que, bien que moins intenses, ces erreurs locales sont toujours présentes. Cela s'explique car notre méthode, en effectuant la SVD tronquée sur un échantillonnage censé être représentatif de tout le champ, choisit un modèle minimisant l'erreur moyenne. Les trajectoires peu représentées, n'apparaissant par exemple que dans une petite zone turbulente, seront moins bien décrites par ce modèle. Pour pallier ce problème, on pourrait envisager d'estimer des bases différentes par régions, en effectuant une segmentation spatiale au préalable. Il s'agit d'une perspective dont nous reparlerons en fin de chapitre.

Comparaison des estimations avec base polynomiale et base apprise

Les estimations obtenues par FOLKI-MF avec la base polynomiale (ce qui correspond à LKFT) et avec la base SVD isotrope sont présentés dans la partie inférieure de la figure 3.8. Les deux méthodes utilisent le même nombre d'images $N = 8$, un ordre de modèle $d = 3$ (donc polynôme de degré 3 dans un cas et approximation sur 3 vecteurs de base dans l'autre), et une fenêtre carrée uniforme de 33 pixels de côté (c'est-à-dire de "rayon" égal à 16 pixels). Ce choix de paramètres sera justifié dans la section 3.4.2. On retrouve la réduction de biais observée sur l'erreur de modèle, les bouffées locales d'erreurs sont localisées dans les mêmes zones que les fortes erreurs de modèle et sont moins intenses avec la base apprise. Quant au bruit de mesure, il reste faible, comme avec la base polynomiale. Nous avons donc réduit le biais tout en conservant l'avantage d'une variance

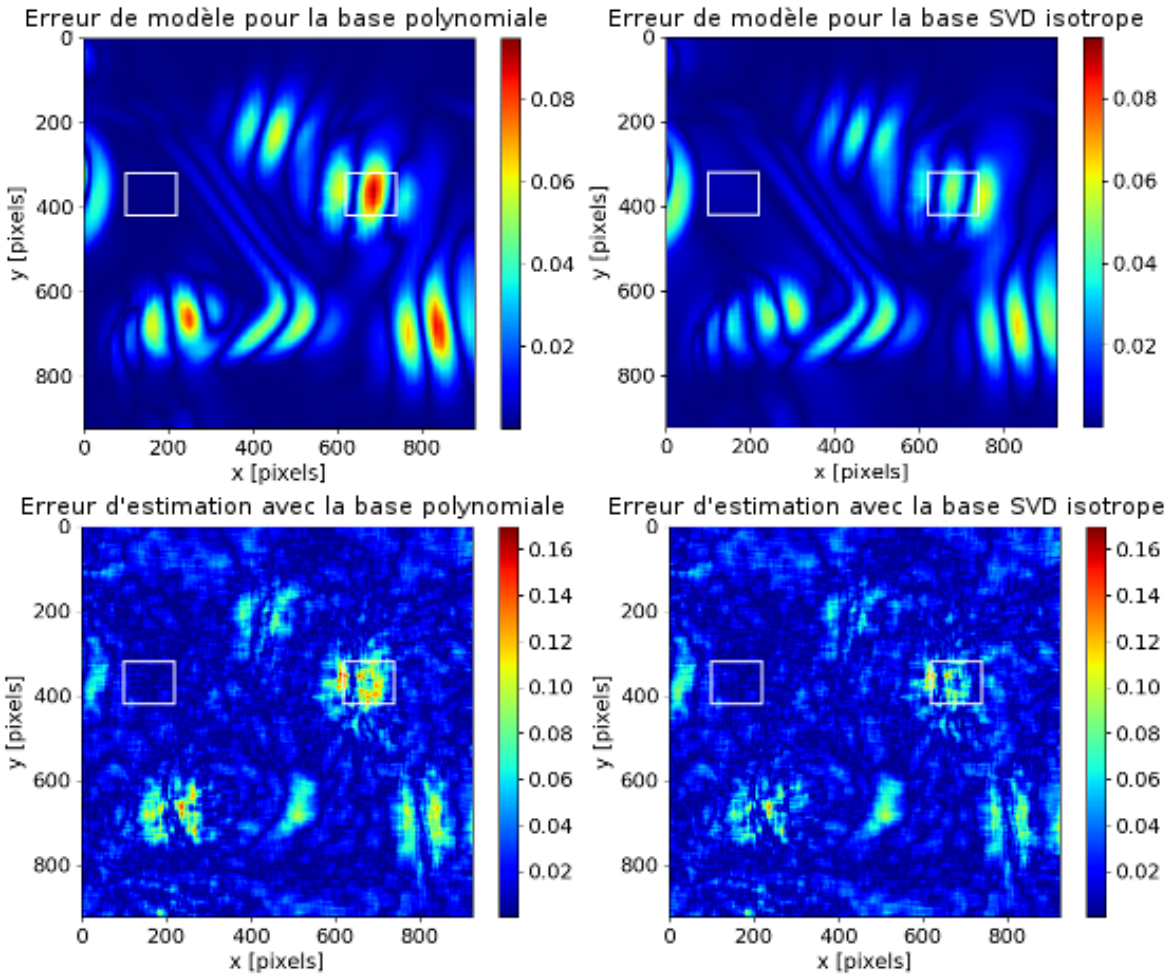

FIGURE 3.8 – Cartes d'erreur (en pixels) du champ transverse de déplacement $v(t_{ref+1})$, de la séquence de l'aile battante. Erreur de modèle en haut et erreur d'estimation en bas (avec pour les deux méthodes $N = 8$, $d = 3$ et une fenêtre carrée uniforme de 33 pixels de côté).

faible. Les résultats étant très proches dans les cas isotrope et asymétrique, nous n'avons représenté que ceux obtenus avec la version isotrope.

Pour étudier plus spécifiquement la robustesse au bruit des méthodes multiframe présentées, on fait varier le niveau du bruit additif gaussien appliqué aux images et on trace l'erreur moyenne et l'erreur maximale (il s'agit de statistiques spatiales pour un instant donné) en fonction de ce niveau de bruit, figure 3.9. Cette expérience montre que l'erreur augmente avec le niveau de bruit, pour toutes les méthodes mais beaucoup plus rapidement pour la méthode biframe. L'erreur maximale obtenue avec les méthodes multiframe est due, comme nous l'avons vu, à l'erreur de modèle. Les courbes de l'erreur maximale montrent qu'à partir d'un certain niveau de bruit, il devient intéressant d'utiliser une méthode multiframe, même dans les zones où celle-ci introduit du biais. Enfin, on peut vérifier sur ces courbes que l'erreur est réduite grâce aux modèles temporels appris, en particulier avec la version "asymétrique" de l'algorithme, utilisant deux bases différentes pour les deux composantes des trajectoires.

Étude paramétrique

On s'intéresse à l'influence de trois paramètres réglables de notre méthode : la taille de la fenêtre, le nombre N d'images consécutives considérées pour l'estimation, l'ordre d

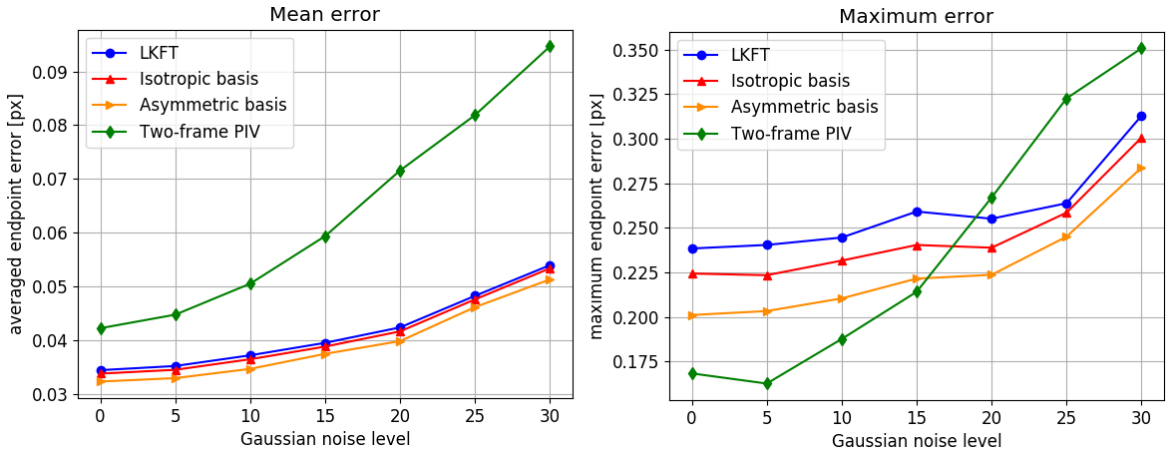

FIGURE 3.9 – Évolution de l'erreur (en pixels) sur le déplacement entre t_{ref} et t_{ref+1} en fonction du niveau de bruit (additif gaussien). Un niveau de bruit de 10 correspond à 4% de l'intensité maximale. L'erreur correspond à la distance euclidienne à la VT, dont on calcule la moyenne ou le maximum spatialement. Pour le maximum on ignore une marge de 50 pixels au niveau des bords. Choix des paramètres : $N = 5$, $d = 2$, fenêtre carrée uniforme de 33 pixels de côté.

du modèle temporel utilisé pour approximer les trajectoires.

La taille de la fenêtre est liée à la régularisation spatiale, qui est plus forte pour une grande taille de fenêtre. Comme le montre la figure 3.10, il existe un optimum pour ce paramètre : une fenêtre trop petite mène à un résultat bruité (le problème étant mal conditionné), tandis qu'une fenêtre trop grande conduit à la perte des structures spatiales fines. C'est ce qui justifie le choix d'une fenêtre de 33 pixels de côté dans les autres expériences.

Les deux autres paramètres influent sur la régularisation temporelle. À d fixé, figure 3.11, augmenter le nombre d'images améliore la réduction de la variance, mais considérer trop d'images augmente le biais, car d est insuffisant pour représenter une complexité accrue. À nombre d'images N fixé, réduire d conduit à régulariser de manière plus importante, ce qui limite la variance mais risque d'ajouter du biais. Réciproquement, augmenter d permet de réduire le biais. Sur la figure 3.10 on remarque que pour une valeur de d plus faible, donc une régularisation temporelle plus forte, il est possible de travailler avec des fenêtres plus petites pour un même niveau d'erreur. On a ainsi plus de chances d'être sensible aux mouvements de structures spatiales fines, autrement dit, on augmente la résolution. Dans l'ensemble, ceci montre que la régularisation temporelle permet de profiter d'une robustesse au bruit sans perdre en résolution spatiale, ou de manière équivalente d'améliorer la résolution spatiale à niveau de bruit fixé. La figure 3.11, graphique de droite, justifie le choix des paramètres $N = 8$ et $d = 3$.

Enfin, attardons-nous un moment sur les résultats obtenus pour des N très grands, figure 3.11. Ces valeurs de N présentent peu d'intérêt en pratique si le but est de réaliser une estimation de déplacement en un instant donné, du moins pour les choix que nous avons faits pour les autres paramètres, dans la mesure où l'on s'éloigne du paramétrage optimal. Dans le cas $d = 2$ on est même bien au dessus du niveau d'erreur de la méthode biframe. Cependant, d'une part c'est là que l'écart se creuse entre les différentes méthodes multiframe étudiées, d'autre part une autre application possible de l'algorithme est d'utiliser les trajectoires estimées sur toute leur longueur, éventuellement avec des valeurs de N élevées. Avec $d = 2$ vecteurs de base, la méthode isotrope donne des résultats très proches de la base polynomiale, et la base asymétrique se démarque en donnant de meilleurs résultats. Avec $d = 3$ vecteurs de base, les méthodes isotrope et asymétrique donnent des résultats semblables, et meilleurs qu'avec la base polynomiale. Le

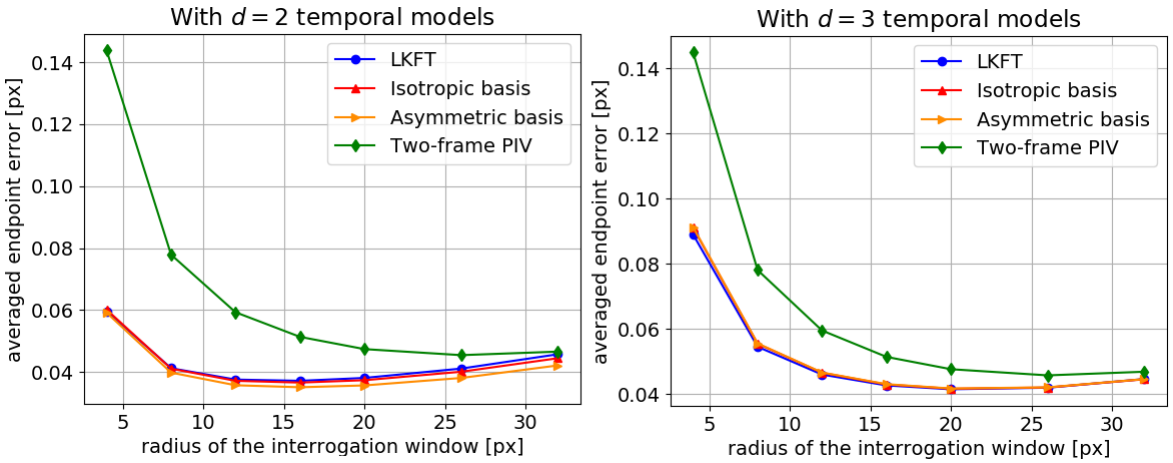

FIGURE 3.10 – Évolution de l'erreur (distance euclidienne à la VT, moyennée spatialement, en pixels) sur le déplacement entre t_{ref} et t_{ref+1} en fonction du rayon de la fenêtre (la fenêtre est uniforme et carrée, de côté $2 \times \text{rayon} + 1$). Le nombre d'images utilisé pour les méthodes multiframe est $N = 5$.

gain d'expressivité conféré par l'approche asymétrique, par rapport à l'approche isotrope, est plus important pour des petites valeurs de d, en supposant qu'il y a effectivement des différences notables entre u et v. Pour $d = 2$, si u et v se comportent très différemment, l'approche isotrope n'aura que 2 vecteurs pour tenter de représenter tous les cas, tandis que l'approche asymétrique en aura 4 en tout, dont deux sont dédiés à u et deux autres à v. Pour $d = 3$, il y a plus de chances pour qu'il y ait des redondances entre les deux bases obtenues avec l'approche asymétrique.

Estimation de trajectoires

Dans les expériences précédentes, nous utilisions une modélisation des trajectoires sur N instants dans le but d'améliorer l'estimation du champ de déplacements entre t_{ref} et t_{ref+1}. Mais l'estimation de trajectoires en tant que telles peut aussi être un but rempli par notre formalisme multiframe. Si le but est d'estimer des trajectoires, on souhaitera sans doute considérer des séquences plus longues que précédemment. Nous allons donc prendre $N = 25$ pour les expériences qui vont suivre, toujours avec $d = 3$. Le but n'étant plus d'améliorer la robustesse au bruit en un instant donné, mais d'évaluer la qualité des trajectoires sur leur longueur. La figure 3.12 compare, dans un cas sans bruit, les trajectoires estimées avec les différentes méthodes aux trajectoires VT. Nous proposons de considérer deux zones du champ, correspondant aux zones encadrées dans les figures 3.5 et 3.8 (on rappelle que celle de gauche est régie par un champ de vitesses lisse, et celle de droite présente des gradients de vitesse élevés). Notre estimation de trajectoires est dense, mais pour des raisons de lisibilité nous n'avons représenté qu'un nombre réduit de trajectoires, sur une grille régulière.

Dans la région présentant de forts gradients de vitesse, première ligne de la figure 3.12, la méthode "SVD isotrope" améliore la justesse des trajectoires estimées de manière significative par rapport à la méthode polynomiale. Cela se voit plus particulièrement sur les trajectoires de la partie gauche de la portion de champ considérée. La méthode "SVD asymétrique" permet une légère amélioration de certaines trajectoires, notamment celle en haut à droite.

Dans la région où la vitesse est plus lisse, deuxième ligne de la figure 3.12, le modèle polynomial donne globalement de meilleurs résultats. Les bases apprises commettent ici

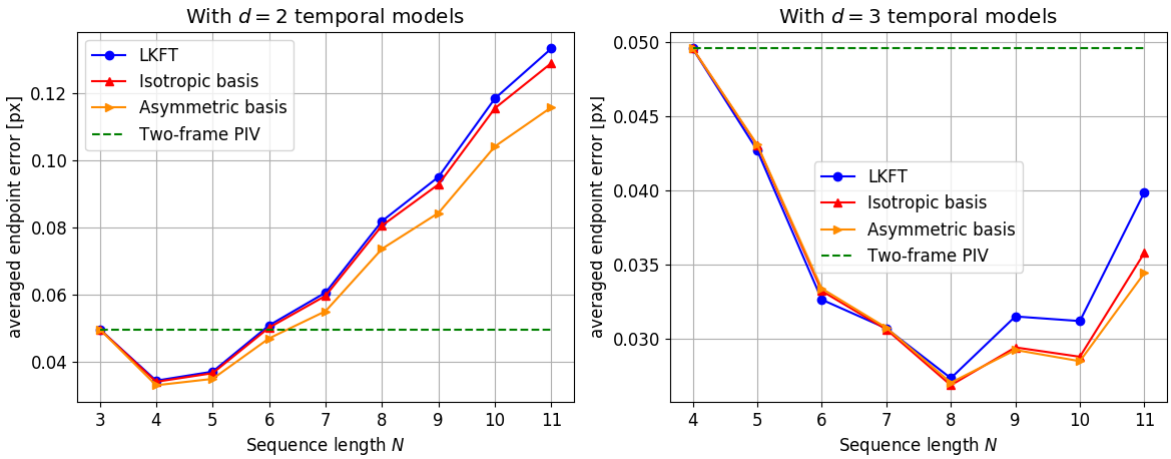

FIGURE 3.11 – Évolution de l'erreur (distance euclidienne à la VT, moyennée spatialement, en pixels) sur le déplacement entre t_{ref} et t_{ref+1} en fonction du nombre N d'images considérées, pour deux nombres d différents de vecteurs de base. Le niveau d'erreur de FOLKI biframe ne dépend pas du nombre d'images, mais est indiqué pour comparaison. Toutes les méthodes utilisent une fenêtre uniforme carrée de 33 pixels de côté.

des erreurs, qui sont légèrement atténuées dans le cas asymétrique. Notre interprétation est la suivante. D'une part la base polynomiale est bien adaptée aux régions calmes, ce qu'on peut justifier par l'approximation de Taylor pour des petits mouvements qui n'est autre qu'une décomposition polynomiale, elle l'est moins pour les régions où les mouvements sont plus violents. D'autre part, la base apprise par SVD tronquée est construite pour minimiser l'erreur moyenne de reconstruction, ce qui se traduit par une homogénéisation des erreurs dans le champ, par rapport à la base polynomiale qui a l'air bien adaptée aux régions calmes. En fait, ici, avec $d = 3$ modèles seulement on n'a pas l'expressivité nécessaire pour représenter à la fois les mouvements des zones calmes et des zones plus agitées. On pourrait donc augmenter d, mais cela signifie diminuer le poids donné à l'*a priori* temporel, y compris dans des régions où cet *a priori* peut être nécessaire pour garantir la qualité de l'estimation. On pourrait également, comme suggéré précédemment, apprendre des bases du mouvement par région.

Perspective : des bases différentes par région

Nous avons évoqué à deux reprises l'idée d'utiliser des bases différentes par région. On pourrait imaginer une segmentation du champ image, par exemple basée sur les gradients d'un champ de vitesses estimé approximativement, via une méthode biframe, éventuellement lissé si le bruit de mesure est gênant. Pour chacune des régions obtenues on effectuerait les opérations décrites précédemment (apprentissage de la base sur un échantillon de trajectoires, puis estimation des coefficients de la décomposition sur cette base grâce à FOLKI-MF). Cela compliquerait l'algorithme dans la mesure où celui-ci repose sur la séparabilité temps-espace qui vient, entre autres, du fait que la base temporelle ne dépend pas de la position spatiale.

Pour éviter cette complication, et conserver le formalisme existant, on pourrait n'utiliser qu'un nombre restreint de bases différentes, disons 2 ou 3, et répéter l'estimation complète pour chacune des bases, puis fusionner les champs obtenus. On conserverait ainsi la séparabilité temps-espace, mais en multipliant les coûts de calcul puisqu'il faudrait faire tourner l'algorithme plusieurs fois. Cela suppose donc, afin de modérer cette augmentation des coûts calculatoires, qu'on puisse se contenter de 2 ou 3 bases différentes, spécia-

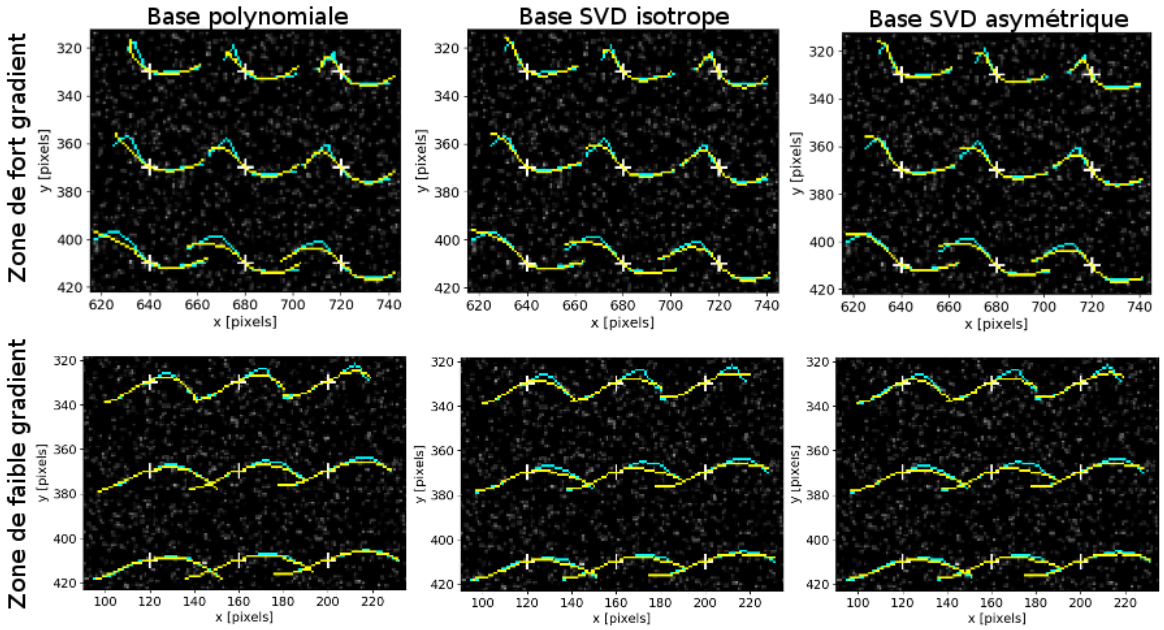

FIGURE 3.12 – Trajectoires estimées (en jaune) comparées à aux trajectoires VT (en bleu), sur la séquence de l'aile battante, pour deux zones du champ présentant des gradients de vitesse forts (première ligne) ou faibles (seconde ligne), ces zones sont encadrées en blanc dans les figures précédentes. Paramètres choisis : $N = 25$, $d = 3$, fenêtre carrée uniforme de 33 pixels de côté.

lisées pour différentes plages d'amplitude des gradients par exemple. La base spécialisée pour les forts gradients serait la même pour toutes les zones où les gradients dépassent un certain seuil. Chacun des champs estimés serait précis dans les zones sur lesquelles la base utilisée est spécialisée, et la fusion pourrait être faite de sorte à minimiser localement une erreur de reconstruction ou un équivalent du score de corrélation calculé à partir du résidu du critère aux moindres carrés de l'algorithme.

3.5 Conclusion du chapitre

Nous avons développé un formalisme permettant l'utilisation de modèles temporels appris pour contraindre une estimation dense de trajectoires. Nous avons montré, dans le domaine de la PIV, qu'utiliser une base de modèles temporels appris, mieux adaptée à la séquence d'intérêt, permettait de réduire le biais de l'estimateur multiframe tout en conservant sa variance faible. Nos expériences ont été réalisées en apprenant cette base à partir de trajectoires issues de la VT de la séquence étudiée, et constituent donc une preuve de concept encore éloignée d'un cas d'application réel. Pour aller plus loin, il faudrait étudier le cas de l'apprentissage de la base soit

— à partir de trajectoires issues d'un suivi de points ;

— à partir de trajectoires issues d'une simulation.

on peut également imaginer apprendre la base à un moment donné de la séquence et s'en servir pour estimer le mouvement à un autre moment, ou encore sur une autre séquence PIV, afin de voir si cette base peut être généralisable ou si elle est spécifique à la séquence dont elle provient.

Nous avons également étudié l'influence des paramètres N et d régissant la régularisation temporelle apportée par notre algorithme. Dans la limite de la cohérence temporelle de la séquence (dépendant notamment de la cadence d'acquisition et des temps

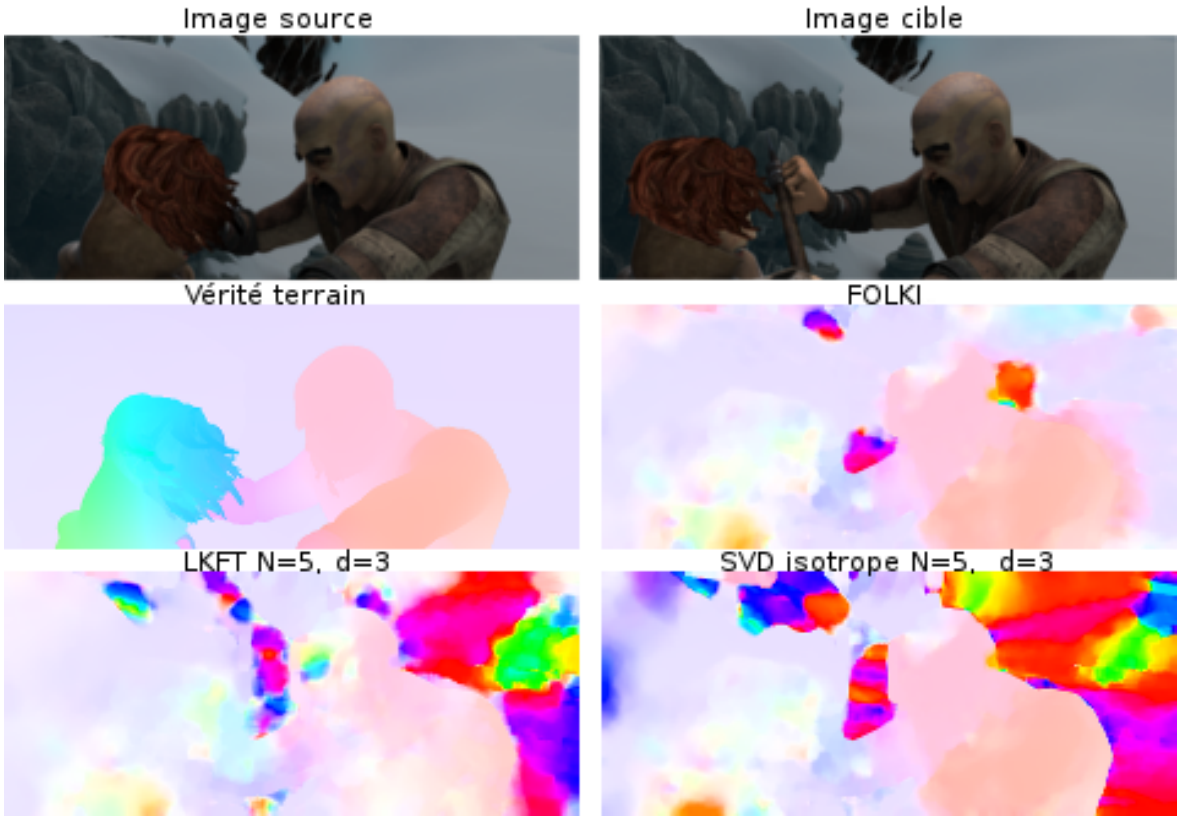

FIGURE 3.13 – Flots optiques estimés par FOLKI et ses extensions multiframe LKFT et FOLKI-MF (avec base apprise par SVD isotrope), sur un exemple de Sintel (ambush5 frame 0036).

caractéristiques des phénomènes observés), augmenter N permet de profiter d'une régularisation temporelle plus forte. Une grande valeur de N suppose cependant que d soit suffisamment grand pour permettre de décrire des trajectoires longues donc potentiellement complexes. Cependant, la régularisation temporelle est plus forte avec d petit. Pour profiter au maximum de la redondance temporelle, il faut donc prendre N aussi grand que la cohérence temporelle de la séquence le permet, et d aussi petit que la complexité des trajectoires le permet.

LKFT et FOLKI-MF permettent, comme nous l'avons constaté, une amélioration des résultats par rapport à la version biframe de FOLKI, en considérant $N > 2$ images, dans le cadre de la PIV. Sur des données de vision par ordinateur, comme celles de MPI Sintel, l'augmentation de l'horizon temporel entraîne, au contraire, une dégradation de l'estimation. On compare, sur la figure 3.13, les flots optiques estimés par FOLKI, LKFT et FOLKI-MF (avec une base obtenue par SVD isotrope) pour une séquence issue de Sintel. Bien que loin d'être excellente, l'estimation biframe (FOLKI) est globalement la moins fausse. LKFT donne un moins bon résultat dans l'ensemble du champ. FOLKI-MF, au contraire, donne des résultats très faux dans certaines zones (notamment à l'arrière plan) mais présente une meilleure estimation du mouvement du personnage de droite, qui apparaît mieux segmenté que dans l'estimation biframe. Il est probable que la base apprise soit bien adaptée au mouvement du personnage de droite, mais pas aux mouvements du reste du champ, qui sont très différents. Ce résultat appuie une perspective déjà évoquée : présegmenter le champ et utiliser des bases différentes, localement adaptées, dans chaque région. On pourrait *a minima* distinguer trois zones (chacun des deux personnages et le fond), extraire une base pour chacune d'elles et réaliser une estimation avec chacune des bases. Cependant, si les méthodes à fenêtre sont préférées pour des applications de

mesure par imagerie, elles sont peu adaptées aux données de vision qui présentent des discontinuités du champ de mouvement aux frontières entre objets, et parfois de grandes zones peu texturées, comme ici en arrière plan. Par ailleurs, les méthodes décrites dans la littérature récente utilisent l'apprentissage profond et obtiennent des performances remarquables en vision. Nous allons donc, dans la suite, explorer la piste d'une méthode multiframe basée sur l'apprentissage profond, en cherchant à l'appliquer à des données de vision par ordinateur.

3.6 Publication

Les travaux présentés dans ce chapitre, sur FOLKI-MF et l'apprentissage d'une base par analyse en composantes principales, ont été publiés et présentés oralement à l'occasion du congrès ISPIV 2019, GODET et collab. [2019].

3.7 Références

CHAMPAGNAT, F., A. PLYER, G. LE BESNERAIS, B. LECLAIRE, S. DAVOUST et Y. LE SANT. 2011, «Fast and accurate PIV computation using highly parallel iterative correlation maximization», *Experiments in fluids*, vol. 50, nº 4, p. 1169. 50, 57, 58

GARG, R., L. PIZARRO, D. RUECKERT et L. AGAPITO. 2010, «Dense multi-frame optic flow for non-rigid objects using subspace constraints», dans *Asian Conference on Computer Vision*, Springer, p. 460–473. 50

GODET, P., F. CHAMPAGNAT, G. LE BESNERAIS et A. PLYER. 2019, «Learning fluid trajectory models for time-resolved PIV», dans *The 13th International Symposium on Particle Image Velocimetry (ISPIV 2019)*. 72

IRANI, M. 2002, «Multi-frame correspondence estimation using subspace constraints», *International Journal of Computer Vision*, vol. 48, nº 3, p. 173–194. 50

JALLAS, D., O. MARQUET et D. FABRE. 2017, «Linear and nonlinear perturbation analysis of the symmetry breaking in time-periodic propulsive wakes», *Physical Review E*, vol. 95, nº 6, p. 063 111. 54

JEON, Y. J., L. CHATELLIER et L. DAVID. 2014, «Fluid trajectory evaluation based on an ensemble-averaged cross-correlation in time-resolved PIV», *Experiments in fluids*, vol. 55, nº 7, p. 1766. 50

LYNCH, K. et F. SCARANO. 2013, «A high-order time-accurate interrogation method for time-resolved PIV», *Measurement Science and Technology*, vol. 24, nº 3, p. 035 305. 50

STANISLAS, M., K. OKAMOTO, C. J. KÄHLER, J. WESTERWEEL et F. SCARANO. 2008, «Main results of the third international PIV challenge», *Experiments in Fluids*, vol. 45, nº 1, p. 27–71. 52, 54

TOMASI, C. et T. KANADE. 1992, «Shape and motion from image streams under orthography : a factorization method», *International Journal of Computer Vision*, vol. 9, nº 2, p. 137–154. 50

YEGAVIAN, R., B. LECLAIRE, F. CHAMPAGNAT, C. ILLOUL et G. LOSFELD. 2016, «Lucas–kanade fluid trajectories for time-resolved PIV», *Measurement science and Technology*, vol. 27, nº 8, p. 084 004. 50, 51, 52, 53, 57, 58

Chapitre 4

Estimation multi-temporelle du mouvement par apprentissage profond

Sommaire

4.1 Introduction

La tendance actuelle en vision par ordinateur, comme dans de nombreux domaines, est d'utiliser l'apprentissage statistique à partir des données et ainsi de se passer du choix du modèle qui est parfois fastidieux. Dans le contexte de la vision, d'excellents résultats ont été obtenus par apprentissage profond, notamment en supervisant l'entraînement de réseaux de neurones convolutifs au moyen de très nombreuses données annotées, pour diverses tâches comme la segmentation sémantique, l'estimation de profondeur, ou plus récemment l'estimation du mouvement. Ces méthodes semblent en particulier présenter une bonne segmentation des objets. En repartant de notre méthode précédente, FOLKI-MF avec apprentissage de modèles temporels par PCA, nous pourrions :

— remplacer l'apprentissage d'une base avec la PCA par un apprentissage profond d'une dépendance temporelle.

— remplacer la brique de mise en correspondance (FOLKI) par une brique réalisant cette tâche par apprentissage profond.

L'espoir d'amélioration est double. D'une part, en supposant les données d'apprentissage suffisamment variées et bien choisies, la modélisation apprise de la dépendance temporelle pourrait être plus générale (on n'aurait donc plus besoin de réapprendre une base à chaque nouvelle séquence étudiée) tout en s'affranchissant du suivi de points. D'autre part, on pourrait s'affranchir des problèmes liés à la fenêtre, et obtenir une structure plus adaptée aux données de vision, les méthodes basées sur des réseaux profonds ayant déjà donné des résultats très satisfaisants au niveau des discontinuités spatiales du mouvement. Par ailleurs, comme nous l'avons vu dans le chapitre 2, ces méthodes sont nettement plus rapides que les méthodes variationnelles.

Dans la logique de l'apprentissage "end-to-end", nous remplaçons toute la chaîne de traitement par un réseau de neurones prenant en entrée N images et donnant en sortie les trajectoires sur les N instants considérés, pour chaque point du champ image. Nous avons vu dans l'état de l'art, en section 2.4.2, qu'il existait quelques méthodes multiframe pour l'estimation de flot optique par apprentissage profond, mais celles-ci ne considèrent que $N = 3$ images consécutives. Nous allons considérer le cas d'un horizon temporel plus long, avec $N = 7$ images. On cherche ainsi à apprendre la tâche complète d'estimation de mouvement avec exploitation de la cohérence temporelle sur un horizon temporel étendu. Pour entraîner ce réseau de manière supervisée, nous avons besoin de séquences d'images annotées de la VT des trajectoires en chaque instant et en chaque pixel. Pour cela, nous synthétisons des séquences 2D d'entraînement et quelques séquences générées selon le même procédé mais distinctes afin de disposer d'un jeu d'évaluation. La méthode obtenue est comparée, sur ce jeu d'évaluation, à différentes méthodes biframe afin de montrer l'apport de la prise en compte de la cohérence temporelle.

Nous envisageons deux formalismes différents pour la prise en compte d'un horizon temporel étendu. Le premier consiste, à partir d'une séquence de N images, à choisir une image de référence et à estimer pour chaque pixel de cette image de référence une trajectoire sur les N instants considérés. Il s'agit d'une estimation dense de trajectoires. Ceci correspond au formalisme du chapitre précédent, et c'est ce que nous considérons dans la première partie de ce chapitre. Dans la seconde partie du chapitre, nous cherchons plutôt à estimer le flot optique pour chaque paire d'images consécutives. On obtient alors une suite de $N-1$ champs de déplacements instantanés. La différence entre ces deux formalismes est illustrée sur la figure 4.1.

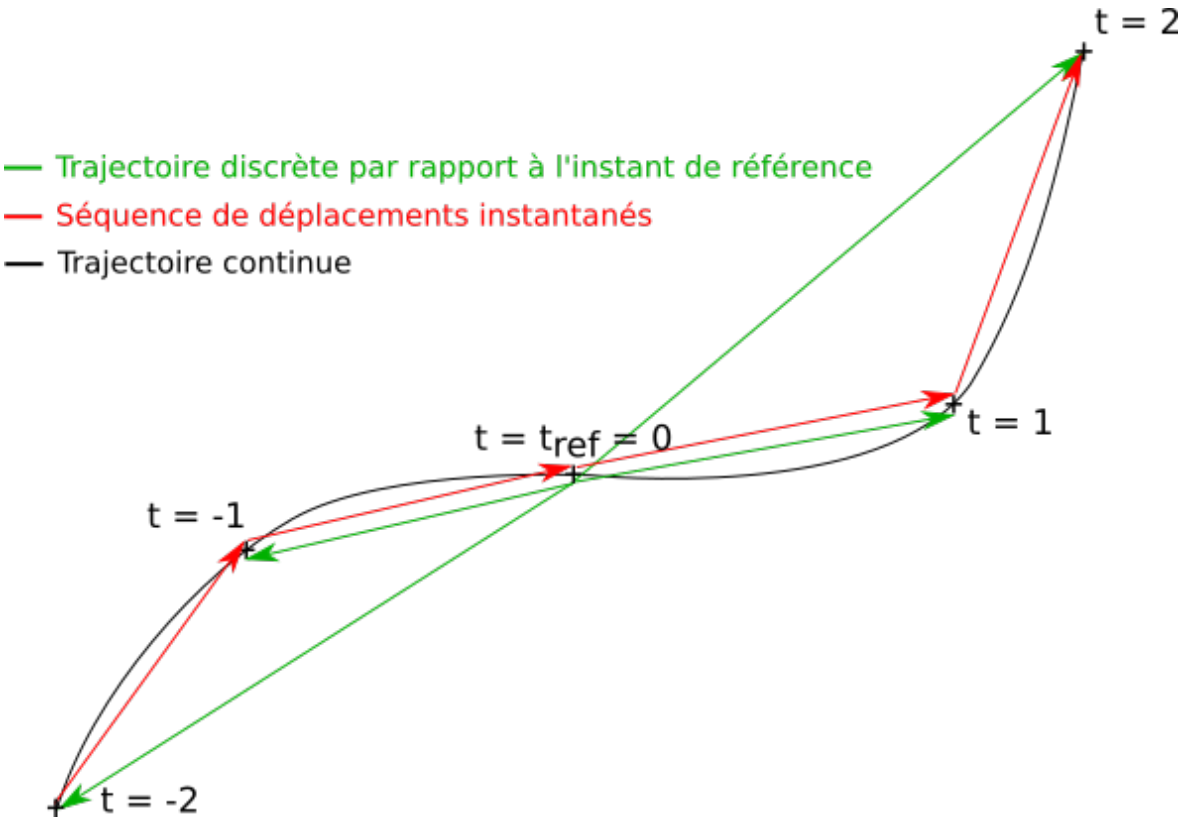

FIGURE 4.1 – Illustration des deux formalismes multiframe, pour un élément spatial, sur une trajectoire de $N = 5$ instants. En vert le fonctionnement en trajectoires, par rapport à l'instant de référence t_{ref}, en rouge le fonctionnement en séquence de déplacements instantanés, ou flots optiques.

4.2 Estimation dense de trajectoires

4.2.1 L'architecture FlowNetStack

Étant donnée une vidéo composée de N images consécutives, la tâche souhaitée consiste à estimer les trajectoires sur N instants, de tous les points situés sur la grille des pixels à un instant pris pour référence. Pour réaliser cette tâche par apprentissage profond, on utilise un réseau de neurones prenant en entrée N images et donnant en sortie les trajectoires sur les N instants considérés. Nous proposons pour cela une architecture basée sur l'architecture FlowNetSimple (DOSOVITSKIY et collab. [2015]).

FlowNetSimple, schématisé sur la figure 4.2, prend en entrée deux images et renvoie le flot optique entre ces deux images. Cette architecture, de type "encodeur-décodeur" est décrite plus précisément en 2.3.2. Ce qui nous intéresse ici est la manière dont est traitée la paire d'images en entrée du réseau. FlowNetSimple ne comporte qu'une seule voie d'entrée pour deux images. Les deux images sont concaténées et le réseau prend en entrée le tenseur résultant de cette concaténation. Pour deux images en couleurs "RVB" (donc à 3 canaux), le tenseur en question aura donc 6 canaux. La première couche convolutive de FlowNetSimple comporte ainsi des noyaux à 6 canaux, et la dernière couche comporte 2 canaux de sortie, pour les composantes horizontale et verticale du flot optique. Entre les deux, dans toutes les couches cachées du réseau, chaque activation peut théoriquement dépendre des deux images d'entrée. Le nombre de canaux de ces couches cachées ne dépend pas du nombre de canaux du tenseur d'entrée, il s'agit simplement du nombre de descripteurs spatio-temporels utilisés. Il est donc possible de changer uniquement le nombre de canaux d'entrée, et donc le nombre d'images concaténées en entrée du réseau, tout en gardant le reste de l'architecture inchangé. De la même manière, on peut changer le nombre de canaux de sortie, sans rien changer d'autre dans l'architecture.

L'architecture "FlowNetStack" que nous proposons est une extension multiframe de FlowNetSimple, qui consiste simplement à changer son nombre de canaux d'entrée pour qu'elle accepte N images concaténées, et son nombre de canaux de sortie pour pouvoir estimer des trajectoires sur N instants, comme représenté sur la figure 4.3. On peut voir le tenseur de sortie comme un empilement de flots optiques, entre une image prise pour référence (au centre de la séquence dans notre cas) et chacune des images de la séquence ; ou bien, de manière équivalente, comme une image de trajectoires : considérant tous les canaux pour un seul pixel, on obtient la trajectoire de ce qui est situé en ce pixel à l'instant de référence. Ceci ne tient pas uniquement à l'architecture, mais surtout à la manière dont elle est entraînée, *ie* à la quantité donnée comme VT, ce qui est détaillé dans la section 4.2.2.

Afin de limiter le nombre de canaux, et ainsi réduire les coûts calculatoires, nous considérons des images en niveaux de gris ayant donc un seul canal. Notre architecture a donc N canaux d'entrée et $2N$ canaux de sorties (déplacements horizontal et vertical en chaque instant). Il est conceptuellement possible d'utiliser des images couleurs, on aurait donc $3N$ canaux d'entrée.

Bien entendu, cette nouvelle architecture doit être complètement réentraînée, les poids de FlowNetSimple n'ayant plus de sens dès lors qu'on mélange, dans les couches cachées, l'information de N images au lieu de 2, et qu'on souhaite estimer des trajectoires sur N instants au lieu d'un flot entre deux instants.

Cet entraînement, supervisé, nécessite des séquences annotées de la VT du mouvement en chaque instant, et une fonction de coût multiframe. Nous obtenons une fonction de coût multiframe en sommant, sur les différents instants, la fonction de coût biframe de

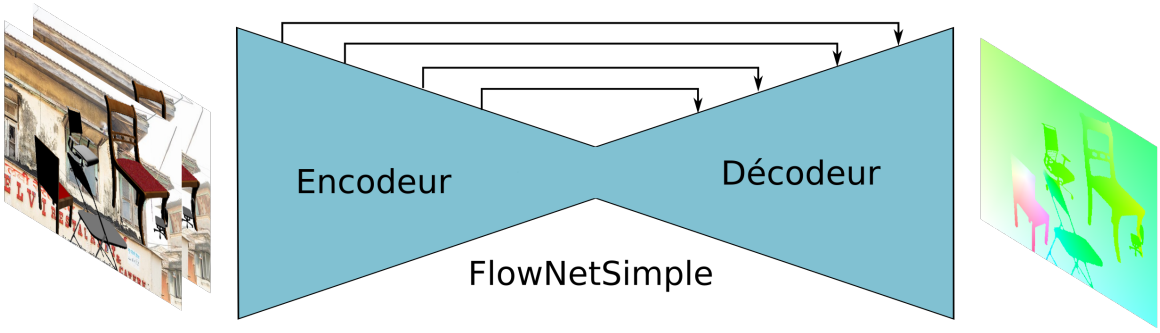

FIGURE 4.2 – Principe de l'architecture FlowNetSimple (DOSOVITSKIY et collab. [2015]).

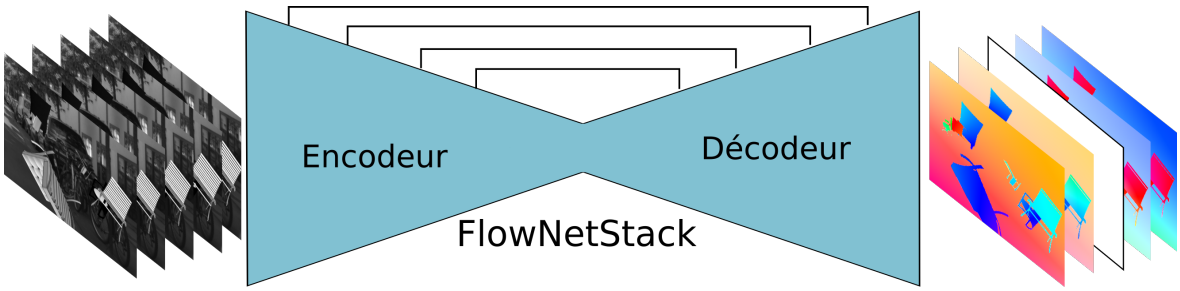

FIGURE 4.3 – Principe de notre architecture FlowNetStack.

DOSOVITSKIY et collab. [2015], basée sur l'erreur *endpoint*, énoncée dans l'équation (2.30). Le jeu d'entraînement proposé par DOSOVITSKIY et collab. [2015], FlyingChairs, ne comporte que des paires d'images et ne peut donc pas être utilisé ici. Les jeux de données FlyingThings3D (MAYER et collab. [2016]) et MPI Sintel (BUTLER et collab. [2012]) comportent des séquences annotées en chaque instant mais cette annotation concerne le flot optique entre images consécutives. Ce qui nous intéresse ici est d'avoir une VT des trajectoires. Nous allons générer notre propre jeu de séquences d'entraînement simulées, décrit dans la section suivante.

4.2.2 Le dataset d'entraînement ChairsMultiframe

Nous souhaitons entraîner notre réseau de sorte à ce qu'il estime les trajectoires, sur N instants, des points situés sur la grille de l'image centrale de la séquence considérée. Nous choisissons ici l'image centrale comme image de référence, afin de limiter l'amplitude des mouvements les plus grands à considérer dans la séquence. Pour réaliser cet entraînement, la VT d'une séquence doit donner pour chaque pixel du champ une trajectoire sur N instants ; ou de manière équivalente, pour chaque instant, le flot optique entre l'instant de référence et l'instant considéré.

Concernant le type de mouvements simulés, il a été montré par ILG et collab. [2017], et confirmé par les expériences de MAYER et collab. [2018], que de meilleures performances étaient obtenues en entraînant sur des mouvements simples avant de présenter des mouvements plus compliqués, nous allons donc, comme dans le cas de FlyingChairs, simuler des mouvements à deux dimensions. D'autre part, le travail de MAYER et collab. [2018] étudie l'influence du type de données d'entraînement sur les performances finales de la méthode, notamment en termes de généralisation à des cas très différents de ceux vus pendant l'entraînement. Les points essentiels caractérisant un bon jeu d'entraînement, permettant une bonne généralisation, sont la diversité des formes et des mouvements représentés. Un bon jeu d'entraînement doit donc être simple mais varié. Utiliser des

chaises permet d'obtenir une grande diversité de formes, et avec des détails d'échelles variées (les pieds des chaises sont souvent des structures spatiales fines, tandis que le dossier peut être très différent d'une chaise à l'autre). Nous proposons donc une version multiframe de FlyingChairs, que nous nommons "ChairsMultiframe".

Dans la mesure où notre but est l'exploitation de la cohérence temporelle, nous allons considérer le cas de séquences à très bon échantillonnage temporel, pour simuler une haute cadence d'acquisition.

Méthode de génération de séquences multiframe

Les images sont générées en apposant des images de chaises sur des images de fonds variées. Au total, 5000 séquences sont générées, avec 33 images par séquence. Afin d'éviter une sur-spécialisation du réseau sur des cas trop simples, il est nécessaire de créer une grande diversité visuelle et représenter une grande variété de mouvements, une génération aléatoire est mise en place pour les différentes composantes des séquences : choix des chaises, choix des fonds, trajectoires des chaises, mouvement d'ensemble du fond. Comme mentionné précédemment, on cherche à générer des données représentatives d'un scénario d'acquisition haute-cadence. En pratique nous allons générer des données telles que l'amplitude des mouvements, entre deux instants consécutifs, soit en moyenne de l'ordre du pixel.

Génération des images

Pour chaque séquence, des images de chaises sont tirées aléatoirement parmi 9744 visuels de ShapeNet (CHANG et collab. [2015]), correspondant à 14 vues 2D de 696 modèles 3D différents. Le nombre de chaises qui apparaîtront dans la séquence est tiré de manière équiprobable entre 2 et 7. Après avoir choisi aléatoirement autant d'images de chaises que le nombre choisi, on applique à chacune d'elles une rotation et une mise à l'échelle aléatoires, de sorte que chaque type de chaise puisse apparaître avec des orientations et des tailles différentes en fonction des séquences. L'angle de rotation est tiré aléatoirement entre 0 et 2π, le facteur d'échelle entre 0.6 et 2.2. Ceci correspond à une taille de chaise (envergure totale) variant de 75 à 275 pixels pour un fond de 640×480 pixels.

Une image de fond est tirée aléatoirement parmi 525 images issues de

— photographies libres de droits disponibles sur Flickr [1], représentant des scènes d'intérieur, des paysages urbains ou des paysages naturels.

— scènes de navigation routière du jeu de données Cityscape (CORDTS et collab. [2016]).

— prises de vues du sol par satellite, du jeu de données xview [2].

Ces images sont de tailles différentes, pouvant aller de 1024×768 à 3000×3000 pixels. L'image choisie est rognée de manière à ne garder qu'une zone, choisie aléatoirement, de taille 640×480 pixels. De manière aléatoire, les images suffisamment grandes sont parfois sous-échantillonnées d'un facteur 2 avant d'être rognées.

Pour chaque instant, des transformations géométriques, avec des paramètres suivant des fonctions continues du temps, sont calculées et appliquées au fond et aux chaises avant que ces dernières soient apposées sur le fond à une position dépendant elle aussi d'une fonction continue du temps. Ces transformations et leur évolution temporelle sont

1. `https://www.flickr.com`, consulté le 19 juillet 2018
2. `http://xviewdataset.org`, consulté le 14 février 2018

FIGURE 4.4 – Quelques images issues des séquences de ChairsMultiframe.

détaillées dans la suite. La figure 4.4 montre quelques exemples d'images extraites de ChairsMultiframe.

Génération des mouvements

Les fonds sont animés par des transformations homographiques définies par les déplacements rectilignes uniformes indépendants des 4 coins. Plus précisément, un vecteur déplacement 2D est tiré aléatoirement pour chacun des 4 coins, le coin correspondant suit un mouvement rectiligne uniforme, sa translation entre deux instants consécutifs est donnée par ce vecteur. Chacune des 8 composantes (déplacements horizontal et vertical de chacun des 4 coins) est tirée indépendamment selon une distribution gaussienne centrée d'écart-type 0.4 pixel. Cette valeur a été choisie de sorte à ce que les mouvements des chaises soient plus rapides que ceux du fond.

Le mouvement des chaises est composé de rotations, mises à l'échelle et translations dont les dépendances temporelles sont des fonctions B-splines. La trajectoire du centre d'une chaise est obtenue en générant d'abord des points clefs, puis en calculant une interpolation B-spline cubique pour chacune des deux coordonnées du plan, de sorte à obtenir une trajectoire passant par ces points. On génère ainsi une trajectoire continue en contraignant assez simplement sa forme globale (notamment sa longueur totale, sa courbure et ses changements de direction). Les points clefs sont générés de la manière suivante :

— on tire aléatoirement le nombre de points clefs entre 4 et 8 inclus, notons K le nombre de points clefs.

— on tire aléatoirement le déplacement moyen entre instants consécutifs, selon une distribution gaussienne de moyenne 1.25 pixels et d'écart-type 0.5 pixels.

— en multipliant ce déplacement moyen entre instants consécutifs par le nombre d'instants on obtient un déplacement cumulé D sur toute la trajectoire. Plus précisément, D est la somme des longueurs des segments reliant les points clefs consécutifs. En réalité, la longueur curviligne totale après interpolation sera un peu plus grande que D, mais cette paramétrisation suffit pour régler l'ordre de grandeur de l'amplitude des mouvements.

— on génère aléatoirement $K-1$ distances $d_1 \dots d_{K-1}$ telles que leur somme soit égale au déplacement cumulé D souhaité. On ajoute une contrainte supplémentaire à cette génération aléatoire : chaque segment doit être au minimum de longueur $\frac{D}{2(K-1)}$.

— une position initiale est tirée aléatoirement dans le champ, selon une distribution uniforme, c'est le premier point clef.

— on tire aléatoirement un angle, indiquant dans quelle direction placer le point clef suivant. Le second point clef est placé à une distance d_1 du premier, dans la direction choisie.

— on répète cette opération, le i-ème point est placé par rapport au précédent, en se déplaçant de la distance d_{i-1}, dans une direction tirée aléatoirement. Cette direction est définie par rapport à la direction précédente, de sorte qu'un angle nul consiste à continuer dans la même direction, et qu'on puisse borner l'angle afin de limiter la courbure ou les changements de direction de la trajectoire. En pratique, la valeur absolue de cet angle est tirée aléatoirement de manière uniforme entre $\frac{\pi}{6}$ et $\frac{\pi}{3}$ son signe est également choisi aléatoirement.

La courbure obtenue dépend essentiellement de l'amplitude des angles et des distances entre points consécutifs, les changements de direction sont liés au nombre de points et au choix des angles, la distance cumulée sur toute la trajectoire (couplée au nombre d'images dans la séquence) caractérise l'amplitude des mouvements. Un exemple de trajectoire, avec les points clefs correspondants, est représenté sur la figure 4.5. La trajectoire (continue) ainsi obtenue est échantillonnée de manière régulière sur N instants afin d'obtenir une position par instant discret.

Si une chaise est initialement près du bord et que sa trajectoire se dirige vers l'extérieur du champ, cette chaise va rapidement sortir de l'image. Une telle sortie de champ peut se produire en pratique, ce cas n'est donc pas à exclure. Cependant nous avons choisi de réduire la probabilité d'occurrence de ce phénomène afin d'éviter de générer des séquences se réduisant quasiment au mouvement du fond, toutes les chaises étant sorties rapidement du champ. Pour cela, dès lors que la position initiale est plus proche d'un bord du champ que du centre de l'image, on tire la direction initiale selon une distribution triangulaire maximisant la probabilité d'aller vers le centre et minimisant celle d'aller à l'opposé (entre les deux, la probabilité décroît linéairement). Si, au contraire, la position initiale est plus proche du centre, tous les angles entre 0 et 2π sont équiprobables.

L'angle de rotation et le facteur d'échelle suivent également des fonctions continues du temps qui sont discrétisées aux N instants d'intérêt. La méthode de génération de ces fonctions utilise également des points clefs mais dans une version à une dimension :

— on choisit un nombre K' de points clefs, aléatoirement entre 2 et 4 inclus, et on définit leurs abscisses comme une suite arithmétique entre 0 et 1. L'axe des abscisses correspondra à l'axe temporel (la fonction continue obtenue sera discrétisée sur N instants).

— les ordonnées correspondent à l'angle de rotation, ou au facteur d'échelle, et sont tirées aléatoirement. Afin que les chaises tournent (ou changent de taille) progressivement, ces ordonnées sont tirées selon une distribution uniforme bornée dont la borne est croissante en fonction du temps (*ie* de l'abscisse). L'angle maximal autorisé par rapport à l'orientation initiale (qui ne peut donc être atteint qu'au dernier instant) est $\frac{\pi}{9}$. Quant à la taille des chaises, elle peut varier au maximum de 20 % de la taille initiale.

— on calcule ensuite l'interpolation B-spline cubique uniforme passant par les points abscisse/ordonnée ainsi générés.

La VT de chaque séquence est générée en prenant l'image centrale pour référence. Le champ de déplacements à l'instant central est donc nul. Le champ de déplacements de n'importe quel autre instant donne la position des points de cette image de référence à l'instant considéré.

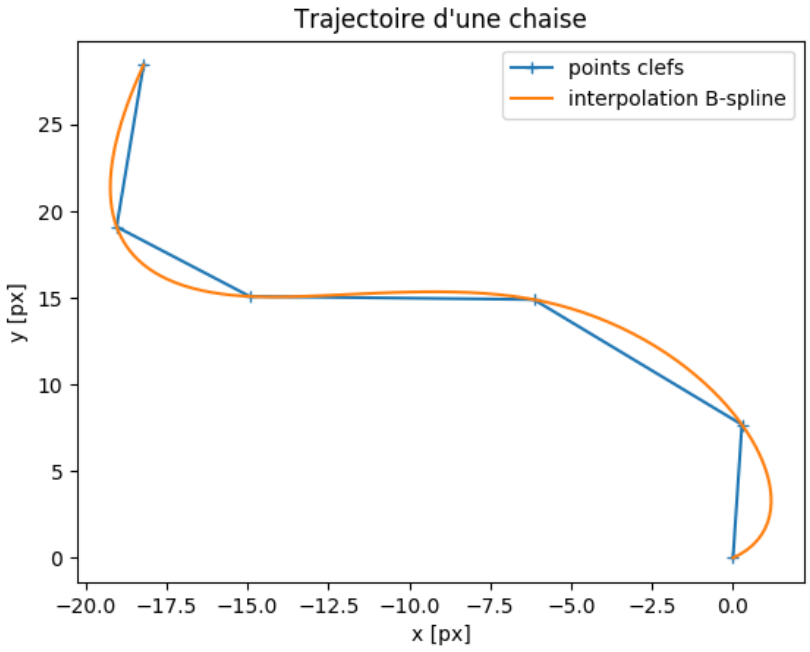

FIGURE 4.5 – Exemple de trajectoire générée pour une chaise.

Afin de limiter l'occupation mémoire de ce jeu de donnée, nous avons appliqué un sous-échantillonnage spatial d'un facteur 2 à toutes les images obtenues. Toutes les statistiques sur les amplitudes de mouvement sont donc divisées par 2. Dans la suite, nous retrouverons des mouvements d'amplitude pixellique, comme prévu initialement, en diminuant la cadence temporelle, en prenant par exemple une image sur deux.

Jeux d'entraînement et de validation

En plus de ces nombreuses séquences d'entraînement, un jeu de 12 séquences de validation est généré selon le même procédé, mais en utilisant des images de fonds et de chaises différentes, qui n'auront pas été vues pendant l'entraînement. Les mouvements sont différents mais générés suivant les mêmes distributions statistiques. Ce jeu de validation est appelé "ChairsMultiframeVal".

Ce jeu de validation est décliné en deux versions. La première, appelée "clean", est du même aspect que les séquences d'entraînement. La seconde, appelée "final", présente une qualité image dégradée par l'ajout de bruit et de flou. Les mêmes 12 séquences sont déclinées en "clean" et en "final", du point de vue de la VT nous avons donc 12 séquences, mais du point de vue des images nous en avons 24 (12 en "clean" et 12 en "final").

Pour générer la version "final", on ajoute un bruit gaussien d'écart-type aléatoirement choisi parmi 1, 3, ou 5 (les intensités des pixels étant codés sur 256 valeurs), et soit un flou de défocalisation, soit un flou de bougé. Le flou de défocalisation est un flou gaussien d'écart-type égal à 1 ou 3. Le flou de bougé est simulé approximativement par un flou gaussien asymétrique : une direction est tirée entre 0 et 2π selon une distribution uniforme, le noyau gaussien asymétrique ayant un écart-type égal à 3 ou 5 dans cette direction, et un écart-type unitaire dans la direction orthogonale. La figure 4.6 met en regard les versions "clean" et "final" d'une même image.

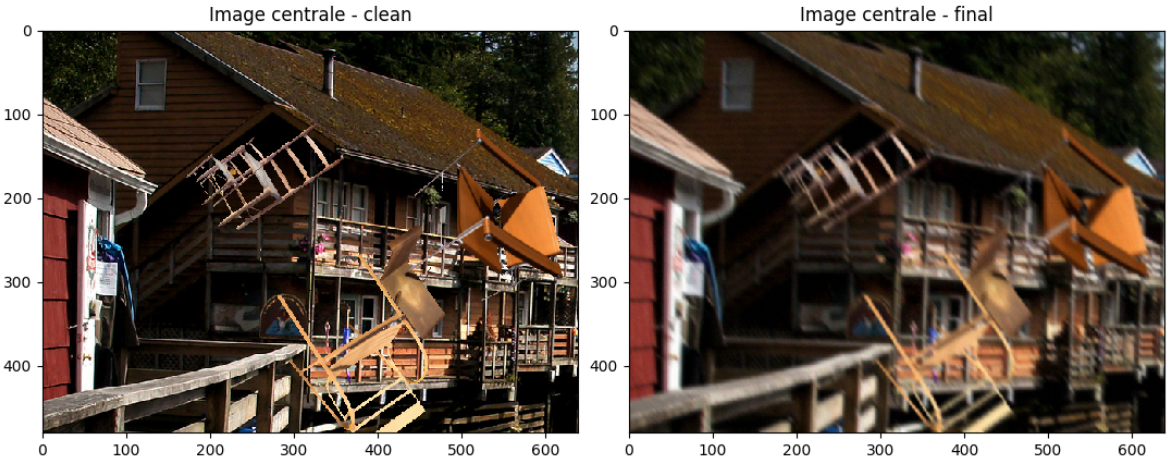

FIGURE 4.6 – Une image de l'ensemble de validation de ChairsMultiframe, en versions clean (à gauche) et final (à droite). Pour obtenir la version "final", un bruit additif gaussien et un flou gaussien asymétrique ont été appliqués à l'image "clean".

4.2.3 Entraînement

Nous entraînons FlowNetStack sur ChairsMultiframe, en utilisant la procédure d'entraînement proposée par DOSOVITSKIY et collab. [2015]. Les seules différences sont l'architecture du réseau, les données d'entraînement et l'ajout d'une somme sur tous les instants dans la fonction de coût.

Pour cet entraînement, nous n'utilisons pas l'intégralité des 33 images disponibles par séquence mais uniquement 7 images, en conservant la même image de référence au centre (la 17ème sur 33) et en sous-échantillonnant temporellement d'un facteur 5. On prend donc l'image centrale et 3 images de chaque côté, en ne gardant qu'une image sur 5, ce qui donne une séquence composée des images 2, 7, 12, 17, 22, 27 et 32. On peut voir en comparant les histogrammes de la figure 4.7 qu'en sous-échantillonnant temporellement d'un facteur 5, on étale les amplitudes des mouvements vues lors de l'entraînement, celles-ci restant de l'ordre de grandeur d'un ou quelques pixels.

4.2.4 Améliorations par rapport à l'estimation biframe

Nous avons vu dans le chapitre précédent qu'une méthode multiframe pouvait permettre d'améliorer l'estimation du champ de déplacements entre deux instants, par rapport aux estimateurs biframe. Nous allons ici nous intéresser, comme dans le chapitre précédent, au champ de déplacements entre l'image centrale et celle qui la suit. Dans notre cas, où nous considérons un horizon temporel de 7 images, il s'agit de la paire composée des 4ème et 5ème images.

Nous allons comparer les résultats de notre méthode FlowNetStack, entraînée comme expliqué ci-dessus, avec les architectures biframe FlowNetSimple (DOSOVITSKIY et collab. [2015]) et PWC-Net (SUN et collab. [2018]) entraînées sur ChairsMultiframe dans les mêmes conditions. Pour les architectures biframe, on ne donne que deux images par séquence en entrée du réseau. Nous avons considéré deux possibilités : utiliser la paire d'images centrale "center" ou la paire d'images extrême "end". Pour chaque séquence de 7 images, une méthode "center" voit la 4ème et la 5ème, tandis qu'une méthode "end" voit la 4ème et la 7ème. On rappelle que la 4ème image est nécessairement impliquée, car il s'agit de l'instant de référence par rapport auquel toutes les vérités terrains sont générées.

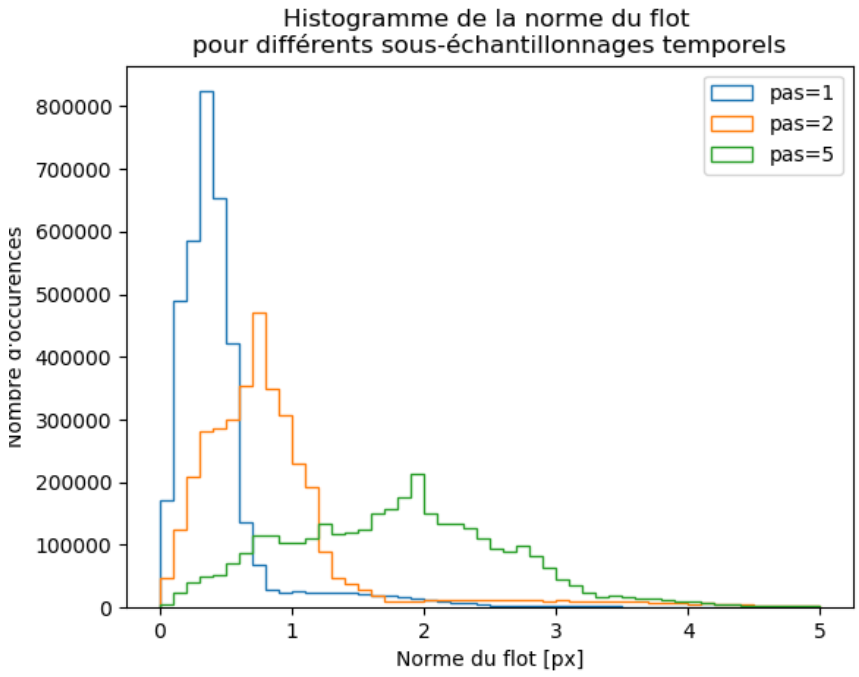

FIGURE 4.7 – Histogramme de la norme du flot entre deux instants consécutifs, pour différents sous-échantillonnages temporels, pour l'ensemble de validation de ChairsMultiframe.

Toutes les méthodes sont entraînées en utilisant la version de ChairsMultiframe sous-échantillonnée d'un facteur 5. Pour l'évaluation, on pourra également utiliser la version sous-échantillonnée d'un facteur 2, ou celle sans sous-échantillonnage. Comme vu précédemment sur la figure 4.7, les mouvements y sont globalement sous-pixelliques, ceci permettra d'évaluer la sensibilité des différentes méthodes à ce type de petits mouvements.

Deux mesures d'erreur sont utilisées : l'erreur *endpoint*, définie en section 2.1.5, qui correspond à la distance euclidienne entre le flot estimé et la VT, et l'erreur relative qui est une erreur *endpoint* normalisée localement par la norme du flot VT. Ces deux mesures d'erreur sont moyennées sur tout le champ spatial et sur les 12 séquences de validation. Trois versions de ChairsMultiframeVal sont considérées, avec des sous-échantillonnages temporels différents :

— 1 image sur 5, tableau 4.1, les déplacements sont majoritairement pixelliques.

— 1 image sur 2, tableau 4.2, les déplacements sont majoritairement sous-pixelliques mais d'amplitude proche du pixel.

— pas de sous-échantillonnage, tableau 4.3, les déplacements sont largement sous-pixelliques.

Sur la séquence pixellique (1 image sur 5), tableau 4.1, la méthode multiframe FlowNetStack obtient de meilleurs résultats que la méthode biframe FlowNetSimple dont elle est inspirée. Pour un même choix d'architecture, l'ajout d'images supplémentaires améliore donc la qualité de l'estimation. FlowNetStack et la méthode biframe PWC-Net donnent des résultats équivalents du point de vue de l'erreur *endpoint*, cependant en erreur relative l'écart se creuse en faveur de FlowNetStack. Nous en déduisons que l'estimation de PWC-Net est moins précise dans le cas des mouvements de faible amplitude.

On peut évaluer plus spécifiquement l'estimation des mouvements de faibles ampli-

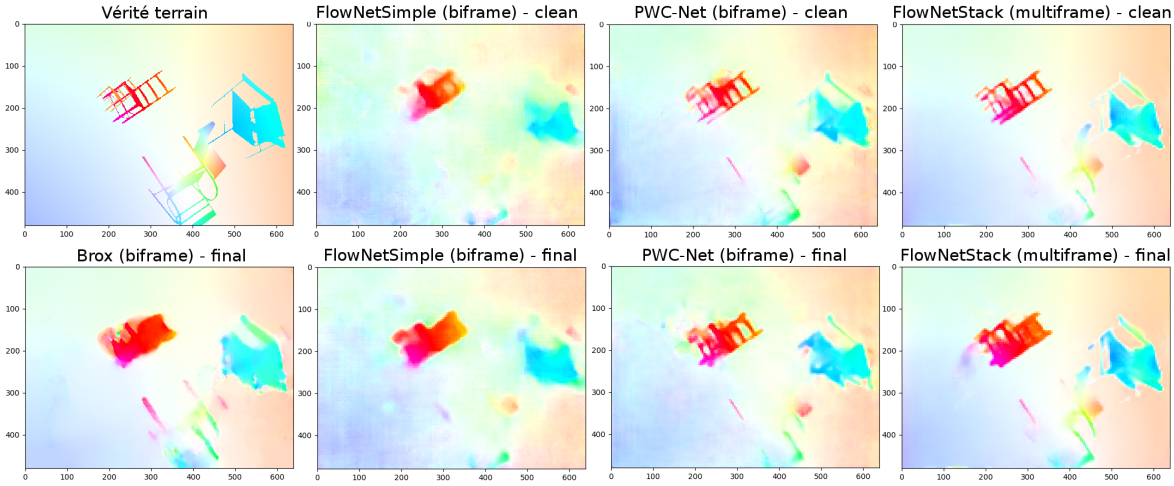

FIGURE 4.8 – Flots optiques estimés sur la version "clean" (en haut) et "final" (en bas) de la première séquence de l'ensemble de validation de ChairsMultiframe, avec sous-échantillonnage d'un facteur 2. En bas à gauche, approche variationnelle de BROX et collab. [2004]. Les méthodes biframe sont celles entraînées en "center".

tudes en comparant les différentes méthodes sur la séquence obtenue en prenant 1 image sur 2, tableau 4.2. L'écart se creuse entre FlowNetStack et PWC-Net, en faveur de FlowNetStack. L'estimation multiframe permet une amélioration remarquable dans le cas des petits mouvements.

Pour la séquence présentant des mouvements encore plus petits, tableau 4.3, l'erreur relative devient démesurément grande, en particulier pour les méthodes biframe. FlowNetStack présente une erreur relative nettement moindre mais tout de même très élevée. Les mouvements étant largement sous-pixelliques ils sont trop petits pour être correctement estimés en ne considérant que deux images consécutives, mais la méthode multiframe peut interpoler en utilisant des instants plus éloignés, et obtenir une estimation approximative du mouvement sous-pixellique.

Nous allons approfondir cette évaluation dans le cas du sous-échantillonnage d'un facteur 2, pour lequel la méthode multiframe se démarque tout en gardant une erreur relative raisonnable. La figure 4.8 présente une comparaison visuelle des champs de déplacements obtenus pour la première séquence de ChairsMultiframeVal dans ce cas. On peut notamment remarquer que les structures spatiales fines ou peu texturées des deux chaises les plus à droite sont mieux estimées avec la méthode multiframe. Le mouvement du fond, très bruité dans les estimations biframe, est nettement mieux estimé par FlowNetStack. Ce mouvement étant stationnaire, on pouvait s'attendre à cette amélioration en augmentant le nombre d'images considérées.

Pour résumer, notre méthode multiframe, associée à une bonne cohérence temporelle des données, est :

— plus robuste que les méthodes biframe aux dégradations de la qualité image comme le bruit ou le flou.

— plus sensible que les méthodes biframe aux mouvements des structures spatiales fines et, ainsi, restitue mieux les discontinuités du champ de mouvement.

— plus précise que les méthodes biframe dans l'estimation de mouvements sous-pixelliques.

TABLEAU 4.1 – Erreurs moyennes sur ChairsMultiframeVal, pour l'instant central, avec un sous-échantillonnage temporel d'un facteur 5 (déplacements pixelliques) :

	Clean		Final	
	endpoint [px]	relative [%]	endpoint [px]	relative [%]
FlowNetSimple "center"	0.83	52.0	1.04	75.5
PWC-Net "center"	0.68	40.3	0.82	58.4
FlowNetSimple "end"	0.77	43.7	1.00	70.0
PWC-Net "end"	**0.65**	38.6	0.85	62.4
FlowNetStack (multiframe)	0.67	**27.4**	**0.78**	**46.4**

TABLEAU 4.2 – Erreurs moyennes sur ChairsMultiframeVal, pour l'instant central, avec un sous-échantillonnage temporel d'un facteur 2 (déplacements légèrement sous-pixelliques) :

	Clean		Final	
	endpoint [px]	relative [%]	endpoint [px]	relative [%]
FlowNetSimple "center"	0.40	59.9	0.51	86.4
PWC-Net "center"	0.34	47.7	0.44	77.0
FlowNetSimple "end"	0.45	67.8	0.58	96.2
PWC-Net "end"	0.37	60.2	0.49	88.6
FlowNetStack (multiframe)	**0.24**	**24.9**	**0.30**	**48.6**

TABLEAU 4.3 – Erreurs moyennes sur ChairsMultiframeVal, pour l'instant central, sans sous-échantillonnage temporel (déplacements largement sous-pixelliques) :

	Clean		Final	
	endpoint [px]	relative [%]	endpoint [px]	relative [%]
FlowNetSimple "center"	0.28	89.5	0.36	112.7
PWC-Net "center"	0.22	68.8	0.30	105.0
FlowNetSimple "end"	0.31	96.5	0.40	128.5
PWC-Net "end"	0.26	84.1	0.35	119.7
FlowNetStack (multiframe)	**0.15**	**36.8**	**0.20**	**71.8**

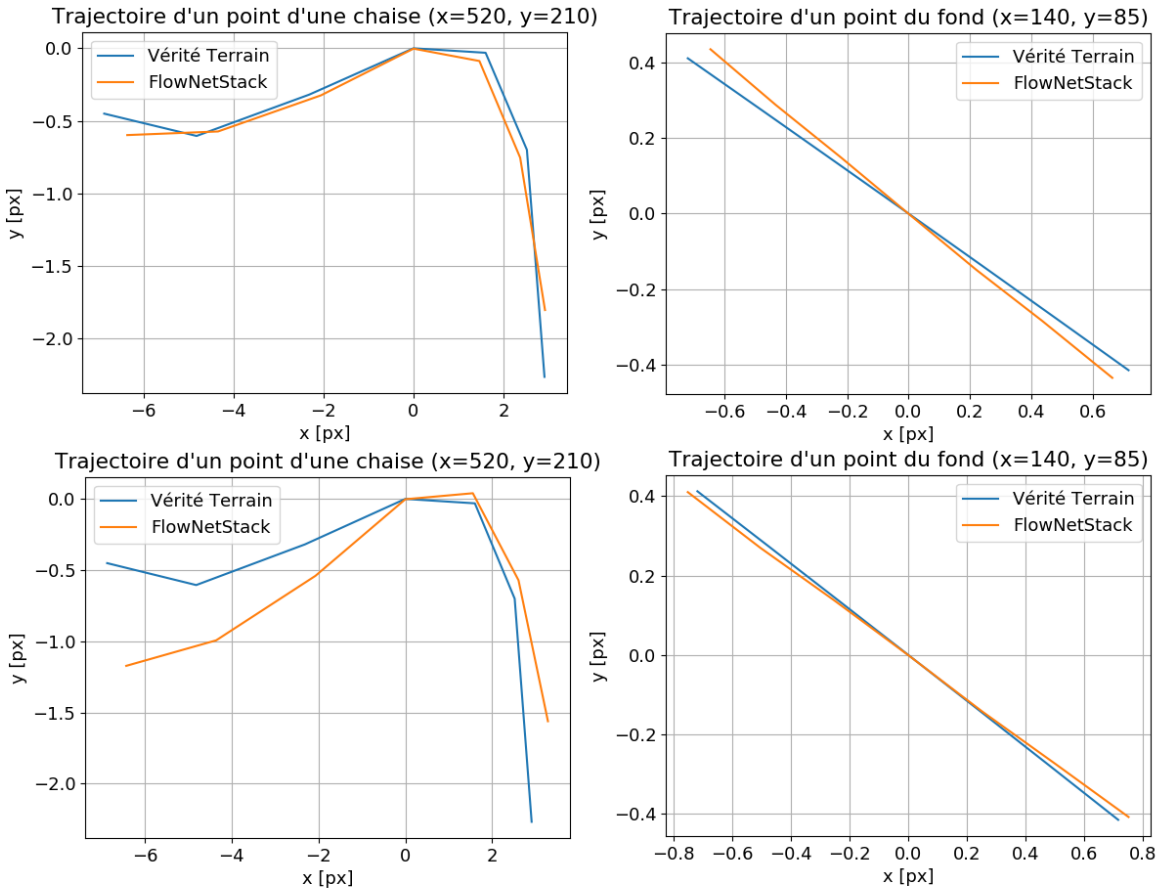

FIGURE 4.9 – Trajectoires estimées par FlowNetStack, sur la première séquence de ChairsMultiframeVal, avec sous-échantillonnage d'un facteur 2, en version "clean" en haut et "final" en bas.

4.2.5 Évaluation de trajectoires

FlowNetStack estime, pour chaque point de la grille de l'image de référence, une trajectoire sur tout l'horizon temporel. Jusqu'ici nous avons considéré le champ de déplacements entre l'image de référence et l'image suivante, nous considérons maintenant la trajectoire dans son ensemble, pour des points particuliers. La figure 4.9 compare la trajectoire estimée par FlowNetStack à la trajectoire VT, pour un point du fond d'une part et un point d'une chaise d'autre part. Le point du fond suit une trajectoire rectiligne uniforme, qui est bien estimée par FlowNetStack, pour les deux versions "clean" et "final" de la séquence. La direction de la trajectoire est légèrement erronée, mais il s'agit bien d'une droite. Le point de la chaise suit une trajectoire plus complexe, qui est assez bien estimée dans la version "clean", bien qu'aux extrémités de la trajectoire l'amplitude du mouvement soit sous-estimée. L'estimation de cette trajectoire est moins bonne en version finale mais reste d'allure semblable à la trajectoire VT.

Bien qu'améliorables, ces résultats permettent de vérifier que l'estimation du réseau a du sens en chaque instant. Ils permettent également une meilleure compréhension de l'amélioration, par rapport aux méthodes biframe, de l'estimation du mouvement du fond, observée précédemment en section 4.2.4 : le réseau semble associer un modèle de droite à la trajectoire du fond, permettant une très bonne régularisation de l'estimation. La légère erreur commise sur la pente de cette droite n'est que peu gênante dans nos expériences de la section 4.2.4, puisque nous considérons un instant proche de l'image de référence.

FIGURE 4.10 – Les 7 images de la séquence du coureur.

4.2.6 Évaluation sur données réelles

Nous évaluons ici nos réseaux entraînés sur les données simulées de ChairsMultiframe, sur une séquence réelle haute cadence (200 Hz), acquise à l'ONERA. Sur cette séquence, un coureur est filmé de profil et suivi par la caméra, on observe donc en particulier un mouvement de défilement du fond et les mouvements des bras et des jambes du coureur. La figure 4.10 montre les 7 images utilisées.

L'évaluation des différentes méthodes sur cette séquence permet de tester leur capacité à généraliser à des données dont les statistiques photométriques sont différentes de celles vues pendant l'entraînement, et mettant en scène des structures autres que des chaises. Cependant, ce cas pratique reste proche des données de ChairsMultiframe sur plusieurs aspects :

— chaque point du fond suit un mouvement rectiligne uniforme.

— les parties du corps peuvent être vues comme des objets mobiles aux trajectoires plus complexes, à l'instar des chaises.

— les mouvements restent dans un plan parallèle à celui du capteur de la caméra.

On ne dispose pas de VT pour cette séquence, nous allons utiliser la méthode variationnelle Deep Flow (WEINZAEPFEL et collab. [2013]) pour obtenir un flot de référence. Nous utilisons l'implémentation disponible dans la bibliothèque OpenCV[3], qui est une version sans l'étape préliminaire de mise en correspondance "Deep Matching" permettant l'estimation des très grands mouvements. Cette étape n'est pas nécessaire ici, la haute cadence d'acquisition garantissant des mouvements d'amplitude suffisamment faible.

La figure 4.11 montre les flots obtenus au niveau de l'instant central de cette séquence, pour les différentes méthodes. Commençons par commenter le résultat de la méthode multiframe FlowNetStack, les mouvements des parties du corps sont bien estimés, la direction et l'amplitude correspondent visuellement à l'estimation obtenue avec Deep Flow, et les discontinuités entre le corps et l'arrière plan sont bien marquées. Cependant, l'amplitude du mouvement du sol est complètement erronée, en particulier la partie la plus proche de la caméra, où les mouvements sont les plus grands. La comparaison des deux variantes "center" et "end" de FlowNetSimple montre une spécialisation du premier sur les petits mouvements (le petit mouvement du poing du coureur est bien estimé, le mouvement plus ample du pied le plus en avant ne l'est pas), et du second sur les mouvements plus grands (c'est l'inverse : le mouvement du poing disparaît tandis que celui du pied est bien estimé). Cela montre un lien fort, auquel on pouvait s'attendre, entre les amplitudes de mouvement vues pendant l'entraînement et le domaine de compétence de la méthode obtenue. Ce lien a aussi été étudié par MAYER et collab. [2018], leurs travaux montrent que les performances obtenues sont très nettement meilleures lorsque la distribution statistique des mouvements est la même entre les jeux d'entraînement et d'évaluation. Cependant il est étrange que FlowNetSimple "center", qui échoue dans l'estimation du grand mouvement du pied, ne soit pas gêné par les grands mouvements du

3. https://opencv.org/

sol, nous y reviendrons. Enfin, parmi les différentes méthodes entraînées, PWC-Net "end" est celle qui donne le résultat le plus propre globalement, mais elle donne un résultat flou sur le haut du corps, jusqu'aux cuisses incluses.

Revenons sur la méthode multiframe FlowNetStack, celle-ci présente un résultat moins flou que celui de PWC-Net sur le haut du corps. Par ailleurs, c'est la seule méthode qui propose une estimation satisfaisante de toutes les parties du corps. Ceci rendrait cette méthode particulièrement intéressante si le sol ne posait pas tant problème. Tentons d'expliquer ce dysfonctionnement. D'une part, la région du sol présentant des mouvements d'amplitude relativement grande est mal estimée, mais le mouvement du pied est d'amplitude aussi grande (comme on peut le constater sur la norme du flot représentée sur la figure 4.12) et est bien estimé. D'autre part, l'algorithme biframe FlowNetSimple "center", spécialisé sur les petits mouvements, estime bien le mouvement du sol mais assez mal le mouvement du pied. Comme FlowNetStack cherche à estimer le mouvement pour l'ensemble des instants, il est probable que le problème vienne des grands mouvements entre instants éloignés (notamment, entre l'instant central et le dernier instant). Le mouvement des points du sol est de plus en plus grand à mesure qu'on s'éloigne de l'image de référence, tandis que le pied finit par arrêter de s'éloigner de sa position initiale pour revenir en arrière. Or pendant l'entraînement, FlowNetStack a vu

— des mouvements relativement grands sur les chaises, mais sur des trajectoires souvent courbées. Il s'agit donc de points qui ne s'éloignent pas indéfiniment de leur position initiale, contrairement aux points du fond qui suivent des trajectoires rectilignes.

— des mouvements rectilignes uniformes des points du fond mais d'amplitude moins grande que les points du sol proches de la caméra dans la séquence du coureur.

Ainsi, FlowNetStack parvient à estimer le mouvement du pied, car il a l'habitude que les objets mobiles puissent bouger relativement vite entre deux instants consécutifs. Mais cette même méthode échoue sur l'estimation du mouvement du sol qui bouge trop vite pour une trajectoire rectiligne et uniforme, et atteint des amplitudes trop grandes en bout de séquence, par rapport à ce qui a été appris.

On peut imaginer plusieurs moyens d'améliorer cette méthode. La première consiste à enrichir la variété des mouvements vus lors de l'entraînement, afin de limiter ce genre de cas de figure inconnus. La seconde consiste à ne plus raisonner en trajectoires mais plutôt sur une série de champs de déplacements instantanés (ou flots optiques) entre instants consécutifs, ainsi on élimine le problème de mouvements devenant de plus en plus amples en bout de séquence.

4.2.7 Conclusion sur l'estimation dense de trajectoires

Nous avons proposé une architecture simple pour prendre en compte $N > 2$ images consécutives dans une méthode d'estimation de mouvement par apprentissage profond. Pour entraîner ce réseau, nous avons généré un jeu de données d'entraînement multiframe, composé de séquences annotées de la VT des trajectoires en chaque pixel et en chaque instant. Cette méthode permet d'obtenir une estimation de trajectoires sur plusieurs instants, mais nous nous intéressons plus particulièrement à la comparaison entre le flot optique obtenu au centre de la fenêtre temporelle (estimé en tenant compte de $N > 2$ images), et l'estimation de ce même flot avec une méthode biframe. Les résultats obtenus sont très encourageants, montrant l'apport du formalisme multiframe dans ce type de méthodes. On pourra notamment citer une meilleure robustesse aux dégrada-

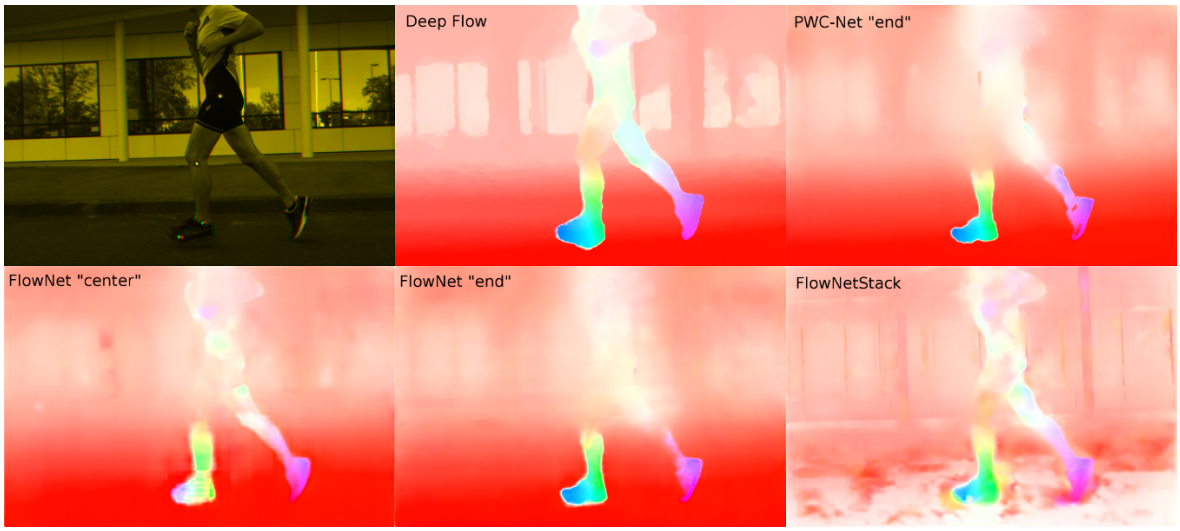

FIGURE 4.11 – Évaluation sur données réelles, sur la séquence du coureur. Flots estimés avec les différentes méthodes entre l'image centrale (4e/7) de la séquence et la suivante (5e/7). En haut à gauche, composition colorée des deux images, la première sur le canal rouge, la deuxième sur le canal vert. Le résultat de la méthode variationnelle Deep Flow est pris comme flot de référence pour la comparaison.

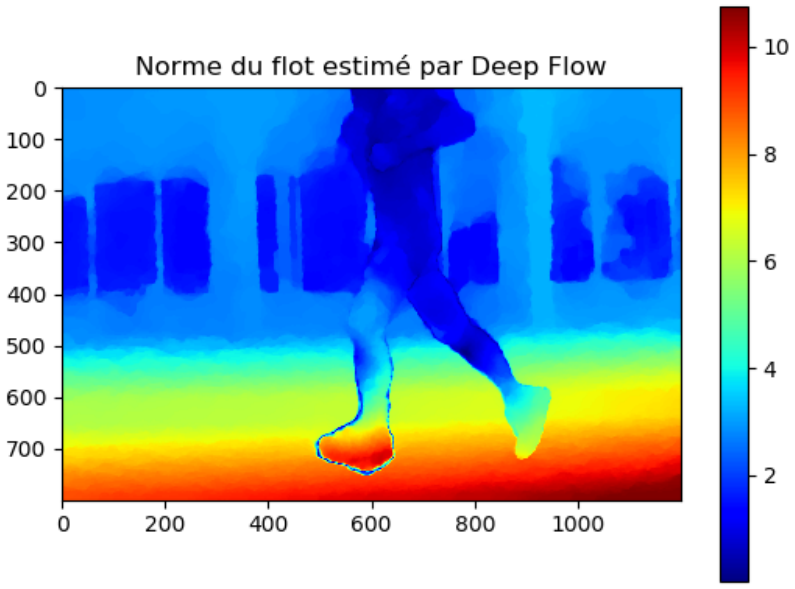

FIGURE 4.12 – Amplitude du mouvement, entre l'image centrale (4e/7) et la suivante (5e/7), pour la séquence du coureur.

tions de la qualité image, une meilleure estimation des structures spatiales fines et une meilleure sensibilité aux très petits mouvements.

Cependant, nous avons mis en évidence un problème de généralisation en évaluant notre méthode sur une séquence réelle. Une explication possible de ce problème est liée à l'existence de mouvements plus grands que ce qui a été vu lors de l'entraînement, en particulier lorsque l'amplitude de ces mouvements est croissante en fonction du temps. Ceci pourrait donc être dû à notre formalisme en trajectoires, c'est-à-dire en estimant les déplacements, à tous les instants, par rapport à un unique instant de référence et non en estimant des déplacements, plus petits, entre instants consécutifs.

4.3 Estimation de champs de déplacements instantanés

Nous proposons d'expérimenter un nouveau formalisme multiframe, basé sur l'estimation de flots optiques entre instants consécutifs. L'idée est donc d'entraîner le réseau à estimer la vitesse instantanée (ou plus précisément les champs de déplacements instantanés) plutôt que les trajectoires. Nous allons également étudier plus précisément l'impact, sur les performances finales, des amplitudes de mouvement vues pendant l'entraînement.

Depuis l'article de DOSOVITSKIY et collab. [2015], plusieurs avancées ont été réalisées dans le domaine de l'estimation de flot optique par apprentissage profond. L'architecture PWC-Net (SUN et collab. [2018]), d'une part, donne de meilleurs résultats que FlowNet, tout en étant beaucoup plus légère que FlowNet2 (ILG et collab. [2017]). Nous allons donc proposer une nouvelle architecture multiframe, basée cette fois-ci sur l'architecture PWC-Net.

D'autre part, les méthodes par apprentissage profond constituent le nouvel état de l'art de l'estimation de flot optique grâce, en grande partie, à un apprentissage en 2 phases : la première sur FlyingChairs, la deuxième sur FlyingThings3D (MAYER et collab. [2016]), un jeu de données simulées mettant en scène des mouvements 3D et d'amplitudes plus importantes, et avec des objets plus variés. FlyingThings propose des séquences d'images annotées, à tous les instants, du flot optique entre instants consécutifs. Il est donc possible de réaliser un entraînement multiframe en utilisant ce jeu de données, dès lors qu'on ne raisonne plus en trajectoires mais en flots consécutifs. Nous expérimenterons donc d'ajouter une deuxième phase d'entraînement, sur FlyingThings3D.

Nous proposons, dans la suite, une comparaison quantitative des différentes méthodes proposées au cours du chapitre sur le jeu de données MPI Sintel (BUTLER et collab. [2012]), proposant des séquences d'images synthétiques mais réalistes, et couvrant une large diversité de mouvements. Ce jeu de données propose des séquences d'images annotées du flot optique entre instants consécutifs, pour tous les instants.

4.3.1 Adaptation du jeu de données ChairsMultiframe

Comme nous voulons maintenant entraîner les réseaux à estimer le déplacement instantané, *ie* le flot optique entre instants consécutifs, nous devons adapter la génération de notre jeu de données d'entraînement afin de disposer de la VT de ces flots entre instants consécutifs. Afin de pouvoir comparer cette nouvelle méthode à la précédente, nous générons, pour chaque séquence, les deux types de VT : les flots entre instants consécutifs et les trajectoires de chaque pixel de l'instant central. Nous générons donc un nouveau jeu de données, appelé ChairsMultiframe2.

Dans les expériences précédentes, avec FlowNetStack, nous utilisions des séquences de 7 images pour l'entraînement, nous fixons donc la longueur des séquences de ce nouveau jeu de données, ChairsMultiframe2, à 7 images. Par ailleurs, les résultats de ces expériences laissent penser que les mouvements présentés pendant l'entraînement étaient d'amplitude trop faible, nous avons donc changé les paramètres des distributions aléatoires régissant la génération des mouvements des chaises et du fond, afin d'autoriser des mouvements de plus grande amplitude. ChairsMultiframe2 est donc généré avec des translations, rotations et mises à l'échelle d'amplitudes plus importantes, les déplacements entre instants consécutifs étant de l'ordre de quelques pixels, voire de la dizaine de pixels. On génère 1800 séquences d'entraînement et 18 séquences de validation. La génération des séquences de validation utilise des fonds et des chaises différents de ceux du jeu d'entraînement.

4.3.2 Architectures de réseaux de neurones multiframe

FlowNetStackInstantané

En reprenant l'architecture FlowNetStack présentée précédemment et en l'entraînant avec la VT des flots entre instants consécutifs, au lieu des trajectoires, on obtient une méthode estimant les déplacements instantanés : pour chaque instant t de la séquence d'images donnée en entrée, le réseau estime le flot entre t et $t+1$.

Nous appellerons cette nouvelle méthode FlowNetStackInstantané. Pour la distinguer, nous appellerons FlowNetStackTraj la méthode précédente, estimant les trajectoires.

Pour être exact, il y a une petite différence d'architecture entre les deux méthodes, résidant dans le nombre de canaux de sortie. Pour N images en entrée, la sortie de FlowNetStackTraj comporte $2N$ canaux, pour représenter une trajectoire 2D sur N instants, tandis que la sortie de FlowNetStackInstantané en comporte $2(N-1)$, correspondant à l'empilement des $N-1$ flots optiques entre instants consécutifs.

Les résultats de ces deux méthodes sont partiellement comparables. FlowNetStackTraj estime les champs de déplacements entre l'instant central t_{ref} et chacun des autres instants. FlowNetStackInstantané estime, pour tout instant t, le flot optique entre t et $t+1$. Donc pour la même séquence d'entrée, les deux méthodes donnent une estimation du flot optique entre t_{ref} et $t_{ref}+1$. Ceci est mis en évidence sur la figure 4.1.

PWCNetStack (Instantané)

L'architecture FlowNetStack tient compte d'une séquence de N images en empilant ces N images dans un tenseur à N canaux qui est pris en entrée du réseau. Nous présentons ici PWCNetStack, une architecture adaptant cette idée au cas de l'architecture biframe PWC-Net (SUN et collab. [2018]). L'architecture PWC-Net est décrite en 2.3.3.

Dans PWC-Net, le même encodeur est utilisé pour générer des descripteurs (appris) pour les deux images d'entrée. La même idée est reprise pour PWCNetStack, chacune des N images passe par un même encodeur qui en donne des descripteurs. La même stratégie multi-échelle est mise en place et, à chaque niveau d'échelle :

— pour chaque paire d'images consécutives, les descripteurs de l'une sont alignés avec ceux de l'autre grâce au flot optique, correspondant à cette paire d'images, estimé à l'échelle précédente ; puis l'opération de corrélation entre ces descripteurs recalés est calculée.

— on concatène, selon les canaux, tous les résultats de corrélation (pour toutes les paires d'images), les flots estimés à l'échelle précédente pour tous les instants, et les descripteurs de l'image centrale.

— le tenseur résultant de cette concaténation est donné en entrée d'un réseau de neurones convolutif ou *Convolutional Neural Network (CNN)* "estimateur de flot", de même architecture que celui de PWC-Net mais dont les nombres de canaux d'entrée et de sortie ont été adaptés. Cet estimateur donne les $N-1$ flots pour l'échelle courante.

On entraîne PWCNetStack en utilisant la VT des flots entre instants consécutifs, comme pour FlowNetStackInstantané.

4.3.3 Le jeu de données FlyingThings3D

FlyingThings3D (MAYER et collab. [2016]) est un jeu de données composé de très nombreuses séquences d'images synthétiques générées à partir de modèles 3D d'objets se déplaçant dans un monde 3D. Contrairement à FlyingChairs (DOSOVITSKIY et collab. [2015]) et à nos différentes versions de ChairsMultiframe, où les mouvements sont uniquement plans, FlyingThings3D présente des mouvements correspondant à la projection, sur un modèle de caméra, de mouvements 3D. Les mouvements sont, aussi, nettement plus violents. Les séquences comportent 9 ou 10 images selon les cas.

Les séquences de FlyingThings3D étant annotées, pour chaque instant, de la VT du flot optique entre instants consécutifs, elles peuvent être utilisées pour l'entraînement de FlowNetStackInstantané et de PWCNetStack.

Il a été montré par ILG et collab. [2017] qu'un entraînement en deux phases, d'abord sur des mouvements simples (FlyingChairs) puis sur des mouvements plus compliqués (FlyingThings3D), donnait de meilleurs résultats qu'un entraînement uniquement sur l'un des deux, ou encore qu'un entraînement sur le mélange des deux. Il s'agit d'une stratégie dite de "curriculum learning". Après avoir comparé les 3 architectures à l'issue d'un entraînement sur ChairsMultiframe2 uniquement, nous pourront considérer une deuxième phase d'entraînement, sur FlyingThings3D, pour les méthodes estimant les flots entre instants consécutifs. Plus précisément, nous utilisons le "subset" de FlyingThings3D, comme expliqué dans la section 2.3.2.

4.3.4 Évaluation sur les séquences de MPI Sintel

Afin d'évaluer et comparer plus précisément les différentes méthodes, nous proposons d'utiliser le jeu de données MPI Sintel, composé de 23 séquences (pour la partie *training* pour laquelle la VT est publique) présentant des mouvements très variés. Nous utilisons la version "Sintel Final", incluant différentes dégradations de la qualité image (flou, effets atmosphériques), rendant les images plus réalistes.

Nous cherchons à répondre à plusieurs questions avec les expériences qui suivent. Dans un premier temps, nous étudions l'impact de l'amplitude des mouvements vus pendant l'entraînement, sur les performances finales de la méthode entraînée. Ensuite, nous étudions les questions du choix du formalisme multiframe (trajectoires ou flots instantanés) et de l'architecture du réseau (FlowNetStack et PWCNetStack). Enfin, nous réalisons une deuxième phase d'entraînement, sur FlyingThings3D subset, et en évaluons l'impact sur les performances du réseau.

Les séquences de Sintel comportent, pour la plupart, 50 images. Afin de réaliser une comparaison quantitative globale sur l'ensemble du jeu de données, nous réalisons

l'estimation sur une fenêtre temporelle glissante de N images, et calculons une erreur moyenne (sur l'ensemble des pixels de tous les flots évalués, et sur l'ensemble des séquences). Comme expliqué en section 4.3.2, le résultat est systématiquement évalué au centre de la fenêtre temporelle (*ie* entre l'instant central et le suivant). On n'évalue donc pas les flots pour les $\frac{N-1}{2}$ (pour N impair) premiers et derniers instants de la séquence de 50 images. Les scores présentés dans la suite ne sont donc comparables ni à ceux présentés dans la littérature, ni à ceux qui seront présentés dans le chapitre suivant, mais sont comparables entre eux.

Nous proposons d'utiliser les métriques du *benchmark* MPI Sintel afin de réaliser une analyse plus fine, notamment selon les amplitudes du mouvement. Ces métriques sont toutes basées sur l'erreur *endpoint* (EPE) moyenne (moyenne spatiale, temporelle, et sur les séquences), mais cette moyenne est réalisée sur des sous-ensembles de pixels pour lesquels le flot VT remplit des conditions différentes. Les différentes métriques sont détaillées ci-après. Dans la suite on s'intéressera tout particulièrement à `all`, `s0-10`, `s10-40` et `s40+`.

— `all` : EPE moyenne sur tous les pixels.

— `noc` : EPE moyenne sur les pixels non occultés seulement.

— `occ` : EPE moyenne sur les pixels occultés seulement.

— `d0-10` : EPE moyenne sur les pixels situés à moins de 10 pixels de la région occultée la plus proche.

— `d10-60` : EPE moyenne sur les pixels situés à une distance comprise entre 10 et 60 pixels de la région occultée la plus proche.

— `d60-140` : EPE moyenne sur les pixels situés à une distance comprise entre 60 et 140 pixels de la région occultée la plus proche.

— `s0-10` : EPE moyenne sur les pixels correspondant à un mouvement d'amplitude inférieure à 10 pixels.

— `s10-40` : EPE moyenne sur les pixels correspondant à un mouvement d'amplitude comprise entre 10 et 40 pixels.

— `s40+` : EPE moyenne sur les pixels correspondant à un mouvement d'amplitude supérieure à 40 pixels.

Influence de l'amplitude des mouvements vus pendant l'entraînement

Avec la méthode FlowNetStackTraj, présentée au début de ce chapitre, nous comparons les résultats en changeant uniquement les amplitudes des mouvements présentées lors de l'entraînement. On rappelle que la nouvelle version de notre jeu d'entraînement, ChairsMultiframe2, se distingue de l'ancienne version, ChairsMultiframe1, par des translations, rotations et homothéties d'amplitudes plus élevées entre instants consécutifs.

Les résultats de cette comparaison sont présentés dans le tableau 4.4. Le modèle entraîné avec ChairsMultiframe2 est nettement meilleur que celui entraîné avec ChairsMultiframe1. Cela suggère que la distribution statistique des mouvements dans ChairsMultiframe2 présente un meilleur recouvrement avec celle de Sintel, les travaux de MAYER et collab. [2018] ayant déjà montré une importante amélioration des résultats lorsque la statistique des mouvements vus pendant l'entraînement correspond à celle des données d'évaluation. En détaillant les résultats suivant les différentes plages d'amplitudes du mouvement, on remarque, dans notre cas, que le gain correspond essentiellement aux mouvements d'amplitudes moyennes (`s10-40`). Le gain est nettement moindre sur les

TABLEAU 4.4 – Erreur moyenne (en pixels) sur Sintel Training Final, pour 2 instances de FlowNetStackTraj entraînées sur des versions différentes de ChairsMultiframe avec des amplitudes de mouvements différentes. Erreur mesurée à l'instant central (entre t_{ref} et $t_{ref}+1$) de la fenêtre temporelle considérée ($N = 7$ images).

Entraînement	all	noc	occ	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
Chairs1	12.00	9.66	37.07	18.07	9.72	5.84	1.10	17.91	76.81
Chairs2	**9.62**	**7.36**	**33.93**	**15.22**	**7.23**	**4.56**	**1.04**	**9.95**	**70.36**

TABLEAU 4.5 – Choix de l'architecture et du formalisme multiframe. Erreur moyenne (en pixels) sur Sintel Training Final. Méthodes entraînées sur ChairsMultiframe2 avec $N = 7$. Erreur mesurée à l'instant central (entre t_{ref} et $t_{ref}+1$) de la fenêtre temporelle considérée ($N = 7$ images)..

Méthode	all	noc	occ	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
FlowNetStackTraj	9.62	7.36	33.93	15.22	7.23	4.56	1.04	9.95	70.36
FlowNetStackInst	9.62	7.35	33.94	15.28	7.22	4.43	1.08	9.99	69.95
PWCNetStack	**8.42**	**6.17**	**32.55**	**13.96**	**5.88**	**3.79**	**0.79**	**6.04**	**68.44**

petits mouvements (`s0-10`) — la méthode entraînée sur ChairsMultiframe1 est spécialisée sur ces petits mouvements, voire sur des mouvements encore plus petits — et les très grands mouvements (`s40+`), pour lesquels les mouvements présentés pendant l'entraînement, même avec ChairsMultiframe2, sont encore d'amplitude insuffisante.

Choix de l'architecture et du formalisme multiframe

Nous comparons maintenant FlowNetStackTraj, utilisant le formalisme en trajectoires, à FlowNetStackInst et PWCNetStack, qui estiment les champs de déplacements instantanés. Rappelons que ces méthodes sont comparables à l'instant central car l'estimation de trajectoire et l'estimation du déplacement instantané sont équivalentes entre t_{ref} et $t_{ref}+1$, comme illustré par la figure 4.1. Pour les trois architectures, les données utilisées pour l'entraînement sont celles de ChairsMultiframe2, pour lesquelles on dispose des annotations aussi bien en trajectoires qu'en déplacements instantanés.

Les résultats sont donnés dans le tableau 4.5. On n'observe que très peu de différence entre les erreurs moyennes obtenues avec FlowNetStackTraj et FlowNetStackInst. Cependant, une nette amélioration est visible avec PWCNetStack par rapport aux deux autres méthodes, sur tous les critères sauf pour les très grands mouvements (`s40+`).

Entraînement sur FlyingThings

Le tableau 4.6 montre l'effet d'une deuxième phase d'entraînement, sur FlyingThings3D, pour l'architecture PWCNetStack. On constate cette fois-ci une très nette amélioration sur les très grands mouvements, une amélioration plus modérée sur les mouvements moyens, et une légère dégradation sur les petits mouvements.

Notons tout de même, au passage, que la comparaison des méthodes au sens de l'erreur *endpoint* `all` incite à être bon sur les grands mouvements : une forte erreur relative sur les petits mouvements n'aura que peu d'impact sur l'erreur moyenne `all`, contrairement à une forte erreur relative sur les grands mouvements. La seconde phase d'entraînement, sur FlyingThings, est donc importante pour obtenir de bons résultats sur le *benchmark* MPI Sintel, mais est à éviter si le contexte de fonctionnement visé implique une précision élevée sur l'estimation de mouvements dont l'amplitude est modérée.

TABLEAU 4.6 – Impact des données vues pendant l'entraînement (Chairs2 = ChairsMultiframe2, Things = FlyingThings3D). Erreur moyenne (en pixels) sur Sintel Training Final, pour l'architecture PWCNetStack avec $N = 7$. Erreur mesurée à l'instant central (entre t_{ref} et $t_{ref} + 1$) de la fenêtre temporelle considérée.

Entraînement	all	noc	occ	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
Chairs2	8.42	6.17	32.55	13.96	5.88	3.79	**0.79**	6.04	68.44
Chairs2 → Things	**5.52**	**3.70**	**25.05**	**10.07**	**3.21**	**2.25**	0.90	**5.80**	**37.95**

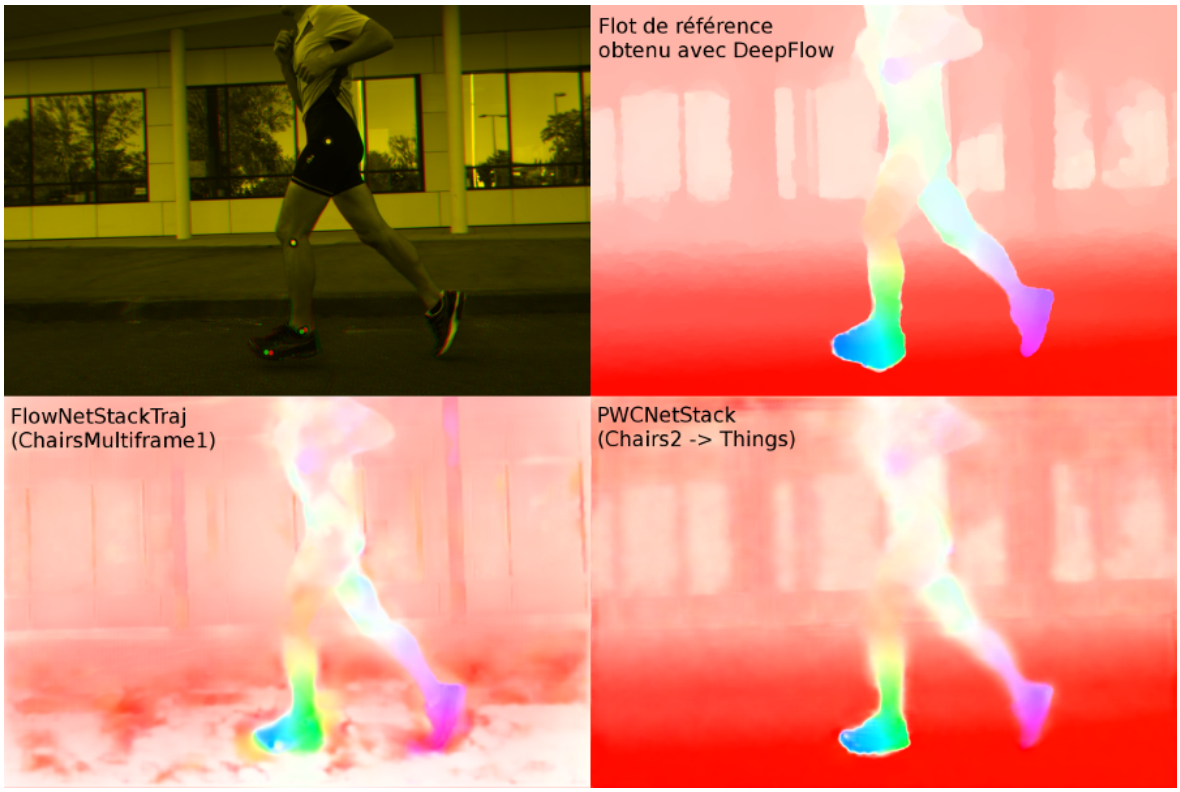

FIGURE 4.13 – Évaluation sur la séquence du coureur, données réelles. Flots estimés avec les différentes méthodes entre l'image centrale ($4^e/7$) de la séquence et la suivante ($5^e/7$). En haut à gauche, composition colorée des deux images, la première sur le canal rouge, la deuxième sur le canal vert. Le résultat de la méthode variationnelle Deep Flow est pris comme flot de référence pour la comparaison.

4.3.5 Discussion

Nous avons montré dans la première partie de ce chapitre que FlowNetStack (version "trajectoires") donnait de meilleurs résultats que les méthodes biframe FlowNet et PWC-Net entraînées sur les mêmes données, dans le cas de l'évaluation sur l'ensemble de validation de ChairsMultiframe. Les expériences sur la séquence du coureur ont cependant révélé des difficultés de FlowNetStack quant à l'estimation de mouvements d'amplitude moyenne (de l'ordre d'une petite dizaine de pixels), ne posant pourtant pas problème aux méthodes biframe. Nous venons de montrer une nette amélioration dans l'estimation des grands mouvements avec notre nouvelle méthode multiframe PWCNetStack et un choix plus judicieux des données d'entraînement (amplitudes des mouvements plus grandes lors de l'entraînement sur les chaises, ajout d'une seconde phase d'entraînement sur FlyingThings3D). On peut vérifier l'amélioration obtenue par rapport à FlowNetStack-Traj sur la séquence du coureur, sur la figure 4.13. Mais comment les résultats de cette

nouvelle méthode PWCNetStack se situent-ils par rapport à ceux de la méthode biframe PWC-Net entraînée sur FlyingChairs puis FlyingThings3D ?

Nous avons réentraîné PWC-Net (biframe), de SUN et collab. [2018], sur FlyingChairs puis sur FlyingThings3D, suivant la procédure établie par ILG et collab. [2017]. Nous obtenons une erreur moyenne sur Sintel (`all`) de 4.1 pixels, ce qui est inférieur aux 5.52 pixels obtenus avec notre meilleure méthode multiframe. En dépit des améliorations notables effectuées par rapport à FlowNetStack entraîné sur ChairsMultiframe1, notre méthode actuelle, PWCNetStack, est moins bonne que PWC-Net biframe sur Sintel. Deux conclusions sont envisageables : soit la continuité temporelle est insuffisante sur ces données pour profiter d'une amélioration notable grâce au multiframe, soit notre méthode n'est pas encore suffisamment bien pensée pour l'exploiter.

4.4 Conclusion du chapitre

Nous avons proposé une méthode multiframe d'estimation du mouvement par apprentissage profond, par empilement des instants successifs, et développé les outils nécessaires pour l'entraîner, en particulier un générateur de séquences annotées. Nous avons expérimenté deux formalismes différents pour l'estimation multiframe : en cherchant à estimer des trajectoires, ayant pour origine les nœuds de la grille d'une image de référence, ou bien en cherchant à estimer une suite de champs de déplacements instantanés.

Nous avons dans un premier temps montré une amélioration par rapport au biframe, dans le cas de dégradations de la qualité image ou de structures spatiales fines, avec FlowNetStack dans le formalisme en trajectoires. Nous avons ensuite constaté des problèmes de généralisation lorsque les mouvements à estimer sont trop différents de ceux vus pendant l'entraînement.

Nous avons ensuite proposé une nouvelle architecture, PWCNetStack, dans le formalisme des déplacements instantanés. En évaluant l'impact des données d'entraînement utilisées sur les performances de notre méthode, nous avons considérablement amélioré les résultats, en particulier sur les grands mouvements, par rapport à notre précédente méthode FlowNetStack.

Si notre nouvelle méthode multiframe, PWCNetStack, est nettement meilleure que notre précédente méthode FlowNetStack, ses performances sur Sintel restent néanmoins inférieures à celles de PWC-Net, méthode biframe sur laquelle elle est basée. D'autre part, notre méthode présente l'inconvénient d'imposer un nombre d'images fixe en entrée, choisi avant même l'entraînement. Une fois l'entraînement réalisé avec $N = 7$ images, l'algorithme ne peut fonctionner qu'avec 7 images d'entrée. Changer N implique de changer le nombre de canaux d'entrée, *ie* le nombre de filtres de convolution de la première couche du réseau, ce qui implique de réentraîner.

Parallèlement à nos travaux, NEORAL et collab. [2018] ont proposé une autre méthode multiframe, ContinualFlow, également basée sur PWC-Net et sur l'estimation de flots optiques entre instants consécutifs. Leur méthode propose un fonctionnement récurrent, en transmettant le flot optique estimé d'un instant au suivant (une entrée supplémentaire reçoit, à chaque instant, le flot estimé à l'instant précédent), permettant l'utilisation d'un nombre d'instants variable. ContinualFlow obtient de meilleurs résultats que PWC-Net sur MPI Sintel. Dans le chapitre suivant nous repartirons des idées de NEORAL et collab. [2018] et proposerons une nouvelle méthode, plus légère et plus performante, permettant de profiter d'une information temporelle à plus long terme.

4.5 Publication

Les travaux présentés dans la première partie de ce chapitre, sur FlowNetStack basé sur un formalisme en trajectoires, ont fait l'objet d'une communication (poster) et d'un papier court (4 pages) à l'occasion du congrès GRETSI 2019, GODET et collab. [2019].

4.6 Références

BROX, T., A. BRUHN, N. PAPENBERG et J. WEICKERT. 2004, «High accuracy optical flow estimation based on a theory for warping», *Computer Vision-ECCV 2004*, p. 25–36. x, 86

BUTLER, D. J., J. WULFF, G. B. STANLEY et M. J. BLACK. 2012, «A naturalistic open source movie for optical flow evaluation», dans *European Conference on Computer Vision*, Springer, p. 611–625. 79, 92

CHANG, A. X., T. FUNKHOUSER, L. GUIBAS, P. HANRAHAN, Q. HUANG, Z. LI, S. SAVARESE, M. SAVVA, S. SONG, H. SU et collab.. 2015, «Shapenet : An information-rich 3D model repository», *arXiv preprint arXiv :1512.03012*. 80

CORDTS, M., M. OMRAN, S. RAMOS, T. REHFELD, M. ENZWEILER, R. BENENSON, U. FRANKE, S. ROTH et B. SCHIELE. 2016, «The cityscapes dataset for semantic urban scene understanding», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 3213–3223. 80

DOSOVITSKIY, A., P. FISCHER, E. ILG, P. HAUSSER, C. HAZIRBAS, V. GOLKOV, P. VAN DER SMAGT, D. CREMERS et T. BROX. 2015, «FlowNet : Learning optical flow with convolutional networks», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 2758–2766. x, 78, 79, 84, 92, 94

GODET, P., A. PLYER, A. BOULCH et G. LE BESNERAIS. 2019, «Estimation de flot optique multiframe par apprentissage profond», dans *XXVIIème Colloque francophone de traitement du signal et des images (GRETSI 2019)*. 99

ILG, E., N. MAYER, T. SAIKIA, M. KEUPER, A. DOSOVITSKIY et T. BROX. 2017, «FlowNet 2.0 : Evolution of optical flow estimation with deep networks», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 6. 79, 92, 94, 98

MAYER, N., E. ILG, P. FISCHER, C. HAZIRBAS, D. CREMERS, A. DOSOVITSKIY et T. BROX. 2018, «What makes good synthetic training data for learning disparity and optical flow estimation?», *International Journal of Computer Vision*, p. 1–19. 79, 89, 95

MAYER, N., E. ILG, P. HAUSSER, P. FISCHER, D. CREMERS, A. DOSOVITSKIY et T. BROX. 2016, «A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4040–4048. 79, 92, 94

NEORAL, M., J. ŠOCHMAN et J. MATAS. 2018, «Continual occlusion and optical flow estimation», dans *Asian Conference on Computer Vision*, Springer, p. 159–174. 98

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018, «PWC-Net : CNNs for optical flow using pyramid, warping, and cost volume», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 8934–8943. 84, 92, 93, 98

WEINZAEPFEL, P., J. REVAUD, Z. HARCHAOUI et C. SCHMID. 2013, «Deepflow : Large displacement optical flow with deep matching», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 1385–1392. 89

Chapitre 5

STaRFlow : Architecture récurrente pour une estimation de flot optique tenant compte du passé

Sommaire

5.1 Introduction

Dans ce chapitre, le but est de développer une méthode d'estimation du flot optique qui tienne compte de manière récurrente de l'information passée, sur un horizon temporel aussi long que possible, afin d'améliorer la qualité d'estimation à l'instant courant. Pour cela, nous repartons des travaux récents présentés dans la littérature sur l'estimation de flot optique par apprentissage profond, établissant l'état de l'art du domaine : PWC-Net (SUN et collab. [2018a]), ContinualFlow (NEORAL et collab. [2018]) et IRR-PWC (HUR et ROTH [2019]). L'architecture PWC-Net sert de base pour de nombreuses autres méthodes, dont ContinualFlow et IRR-PWC.

ContinualFlow est, déjà, une méthode multiframe récurrente profitant de l'information du passé via l'ajout d'une connexion temporelle. À partir de la deuxième paire d'images d'une séquence vidéo, le flot optique estimé à l'instant précédent est transmis en entrée de l'estimateur pour l'estimation courante, comme le montrent les connexions oranges sur le schéma de la figure 5.1. IRR-PWC, fonctionnant en biframe uniquement, propose un principe d'estimation résiduelle itérative, inspiré des méthodes classiques de flot optique, en itérant sur les mêmes opérations apprises aux différents niveaux de l'estimation multi-échelle (les poids des noyaux de convolutions sont les mêmes à toutes les échelles), contrairement à PWC-Net qui utilise des opérations apprises différentes à chaque niveau d'échelle. Ce principe permet une réduction importante du nombre de paramètres du réseau, une grande partie d'entre eux étant réutilisée plusieurs fois. D'autre part, ContinualFlow et IRR-PWC montrent une amélioration remarquable de l'estimation de flot optique grâce à l'estimation conjointe d'une carte d'occultations, et grâce à l'ajout d'un module de *refinement* permettant d'affiner l'estimation notamment proche des discontinuités du champ de mouvement. Du point de vue de l'entraînement, ContinualFlow réalise une première phase en biframe, et sans estimation des occultations, sur le jeu de données FlyingChairs (DOSOVITSKIY et collab. [2015]), puis une seconde, avec la récurrence temporelle et l'estimation des occultations, sur FlyingThings3D (MAYER et collab. [2016]). Pour entraîner IRR-PWC à détecter les occultations dès la première phase, HUR et ROTH [2019] proposent une version de FlyingChairs avec la VT des cartes d'occultations, appelée FlyingChairsOcc. Une deuxième phase d'entraînement est également réalisée sur FlyingThings3D. Avec cette méthode, tout est biframe.

En s'inspirant de ContinualFlow et IRR-PWC, nous repartons du squelette de PWC-Net et construisons une nouvelle architecture récurrente, "STaRFlow" (pour *SpatioTemporal Recurrent Flow*), plus performante et plus légère, en proposant :

— Une nouvelle récurrence temporelle, permettant une mémoire du passé plus générale, ne se restreignant pas au flot optique de l'instant précédent.

— Une récurrence spatiale, comme proposée par IRR-PWC, permettant de partager les poids du réseau pour les différentes échelles de l'estimation. Nous obtenons ainsi une double récurrence spatio-temporelle.

— Une détection des occultations partageant la quasi-totalité des opérations avec l'estimation du flot optique. Ces deux tâches étant fortement liées entre elles, ceci permet de gérer les occultations avec un nombre négligeable de paramètres supplémentaires, pour des performances équivalentes.

Nous obtenons ainsi des performances à l'état de l'art sur les *benchmarks* MPI Sintel et KITTI 2015, tout en proposant une architecture significativement plus légère que les méthodes concurrentes. En particulier, notre connexion temporelle permet une amélioration notable, par rapport à IRR-PWC et ContinualFlow, sur l'estimation des petits objets,

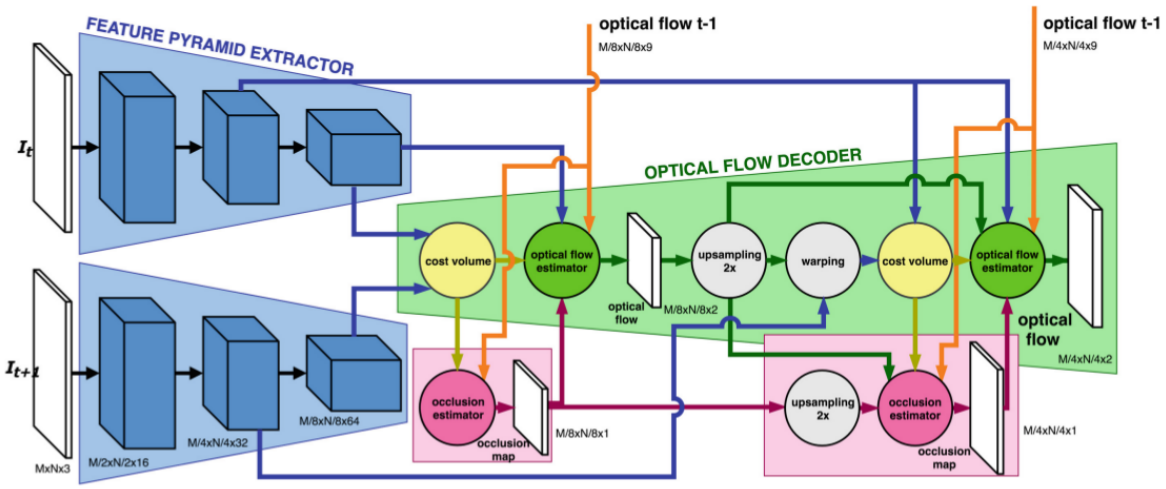

FIGURE 5.1 – Schéma de principe de ContinualFlow. Source : NEORAL et collab. [2018].

des objets partiellement occultés, ou encore lorsque la qualité image est dégradée. Ceci nous permet de dépasser largement ces méthodes sur le *benchmark* MPI Sintel. Sur KITTI, nous dépassons ContinualFlow mais sommes à égalité avec IRR-PWC, sans doute parce que la cadence d'acquisition (10 Hz) est trop faible, et donc les mouvements trop violents, pour profiter d'une cohérence temporelle suffisante.

Nous allons dans un premier temps faire le point sur la façon dont ContinualFlow et IRR-PWC gèrent les occultations et présenter une méthode plus légère et toute aussi performante (section 5.2). Puis, nous présenterons notre architecture (section 5.3) et en particulier notre récurrence temporelle, que nous comparerons à une réimplémentation de celle de ContinualFlow. Ensuite, nous expliquerons la procédure d'entraînement que nous utilisons (section 5.4) pour entraîner les différentes variantes d'architectures présentées. Enfin nous comparerons notre méthode à celles de l'état de l'art sur les *benchmarks* MPI Sintel et KITTI 2015 (section 5.5).

5.2 La question des occultations dans les approches par apprentissage profond

Il a été montré par NEORAL et collab. [2018] et HUR et ROTH [2019] que l'ajout d'un module d'estimation des occultations améliorait la qualité de l'estimation de flot optique, dans un modèle basé sur PWC-Net. Cependant leurs propositions augmentent significativement la taille du modèle. D'autre part, les expériences de ILG et collab. [2018] ne montrent aucune utilité de l'estimation conjointe des occultations, dans le cas d'une architecture basée FlowNet. Cette section a deux buts : vérifier que dans le cas de PWC-Net, servant de base à la construction de notre méthode, l'estimation conjointe des occultations améliore la qualité du flot, et proposer une méthode plus légère pour cela.

5.2.1 Méthodes

Dans ContinualFlow, un module supplémentaire — en rose sur le schéma de la figure 5.1 — composé de 5 couches de convolutions est ajouté à chaque niveau d'échelle pour estimer une carte d'occultations. Ce module prend en entrée le résultat de la corré-

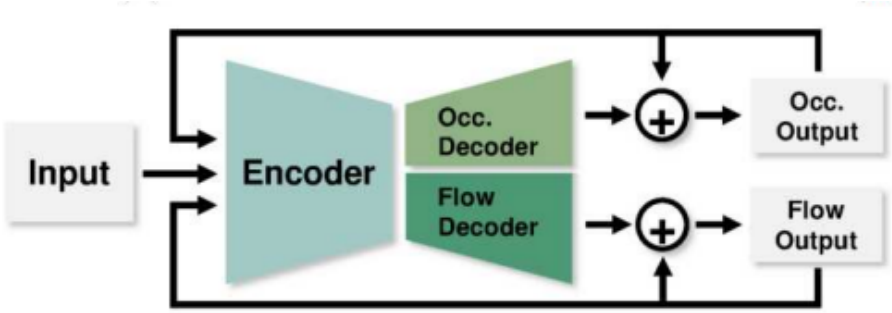

FIGURE 5.2 – Schéma de principe de l'estimation conjointe du flot et des occultations proposée par HUR et ROTH [2019] pour leur architecture IRR-PWC. Source : HUR et ROTH [2019].

lation, et les estimations de flot et d'occultations de l'échelle précédente. La carte d'occultation estimée est passée en entrée de l'estimateur de flot, en plus du résultat de la corrélation, des descripteurs issus de l'encodeur, et du flot de l'échelle précédente. L'ajout de ce module augmente le nombre de paramètres du réseau, ainsi que le temps de calcul, dans la mesure où l'estimation des occultations en entrée de l'estimateur de flot oblige à réaliser ces tâches l'une après l'autre.

Dans IRR-PWC, tout le réseau à l'exception de l'encodeur est dédoublé pour créer une branche d'estimation des occultations, comme représenté sur le schéma de la figure 5.2. Cette branche est donc d'architecture identique à celle du flot, à part le nombre de canaux de sortie qui est de 1 au lieu de 2, et l'ajout d'une fonction *sigmoid* à la sortie. Ceci augmente considérablement le nombre de paramètres du réseau. Contrairement à la méthode de ContinualFlow, ici les deux tâches peuvent être réalisées en parallèle. Si cela peut permettre de réaliser séparément, et donc simultanément, les calculs relatifs aux deux tâches, cela empêche de considérer la carte d'occultations en entrée de l'estimateur de flot optique. Pourtant, les expériences de HUR et ROTH [2019] montrent que les résultats de flot optique sont améliorés en présence de la branche d'estimation des occultations, nous supposons donc que l'encodeur, commun aux deux tâches, apprend dans ce cas des représentations plus générales, et qui permettent une meilleure estimation du flot.

L'analyse de ces résultats, obtenus par HUR et ROTH [2019], indique que les opérations apprises communes aux estimations du flot et des occultations permettent d'obtenir de meilleurs résultats que celles apprises pour l'estimation de flot optique uniquement. Pour aller plus loin dans cette idée, et puisque ces deux tâches sont fortement liées entre elles, nous proposons de partager la quasi-totalité des opérations apprises du réseau pour les deux tâches. Pour cela, nous proposons de réaliser l'estimation conjointe du flot optique et de la carte d'occultations en utilisant un seul décodeur, comme illustré sur la figure 5.3, en ajoutant simplement un canal à la dernière couche de convolution. Après passage dans une fonction *sigmoid*, ce canal supplémentaire donne une carte de probabilité d'occultation avec des valeurs comprises entre 0 (non occulté) et 1 (occulté). Ainsi, pratiquement toutes les opérations apprises du réseau sont partagées pour la réalisation des deux tâches. Seules les convolutions de la dernière couche sont dédiées à une tâche ou à l'autre. Dans la suite, nous appellerons "OccLate" cette manière de réaliser l'estimation conjointe. En comparaison avec les deux méthodes décrites précédemment, ceci conduit à un ajout négligeable de paramètres supplémentaires. De plus nous allons voir, en section suivante, que cette architecture très économe permet d'obtenir des résultats semblables aux propositions précédentes.

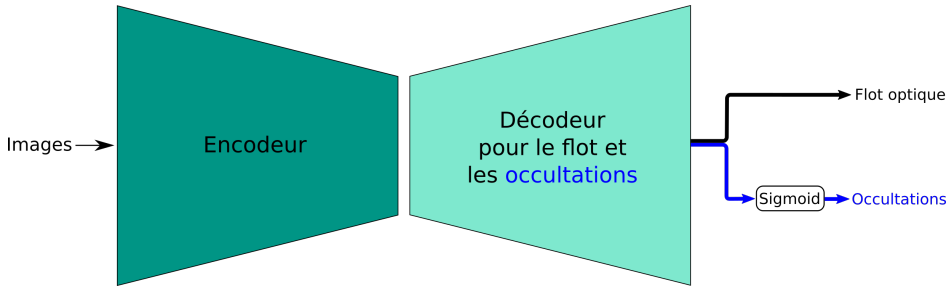

FIGURE 5.3 – Schéma de principe de notre estimation conjointe du flot et des occultations partageant les mêmes opérations apprises.

TABLEAU 5.1 – Comparaison des résultats de flot optique (erreur *endpoint* [px]), en fonction du module d'occultations utilisé, sur les ensembles d'entraînement de MPI Sintel et KITTI 2015. Expériences réalisées en biframe. Les meilleurs résultats sont en caractères gras. Fl-all, sur KITTI, représente le pourcentage d'*outliers* (epe > 3 px).

Modèle	Sintel Clean			Sintel Final			KITTI 2015	
	all	noc	occ	all	noc	occ	epe-all	Fl-all
PWC-Net (réentraîné)	2.74	1.46	16.48	4.18	2.56	21.70	11.75	33.20 %
PWC-Net + OccHur	2.48	**1.30**	15.29	**3.92**	**2.44**	**19.88**	11.31	31.91 %
PWC-Net + OccNeoral	2.52	1.35	15.24	4.04	2.52	20.50	10.73	31.53 %
PWC-Net + OccLate	**2.46**	1.32	**14.82**	3.96	2.47	20.06	**10.58**	**31.28 %**

5.2.2 Expériences et résultats

Repartant de l'architecture PWC-Net, nous développons 3 variantes, implémentant chacun des modules d'estimation d'occultations présentés ci-dessus. Nous réentraînons alors PWC-Net sans module d'occultations et ces trois variantes, selon le schéma habituel FlyingChairsOcc → FlyingThings3D, comme expliqué en section 5.4. Ces expériences sont réalisées dans le cadre d'une estimation biframe, la première phase est donc identique à celle présentée en section 5.4 mais pour la deuxième phase nous n'utilisons que des paires d'images et un taux d'apprentissage initial de 10^{-5}. Nous appelons "PWC-Net + OccNeoral" la variante munie de notre réimplémentation du module d'occultations présenté par NEORAL et collab. [2018], "PWC-Net + OccHur" la variante munie d'une branche d'estimation des occultations d'architecture identique à celle de la branche dédiée au flot à la manière de HUR et ROTH [2019], et enfin "PWC-Net + OccLate" la variante que nous proposons.

Les tableaux 5.1 et 5.2 rassemblent, pour ces différentes variantes, les erreurs obtenues sur l'estimation de flot et sur l'estimation des occultations, ainsi que les nombres de paramètres des architectures obtenues. On peut commencer par remarquer sur le tableau 5.1 que pour les 3 variantes, le fait d'estimer conjointement les occultations conduit à l'amélioration des performances de l'algorithme pour l'estimation du flot optique. Ceci est visible plus qualitativement sur les figures 5.4 et 5.5. Sur l'exemple de la figure 5.4 l'estimation de la direction de l'arme, qui est partiellement occultée dans l'image source et désoccultée dans l'image cible, est améliorée. Sur l'exemple de la figure 5.5 la portion de sol qui se situe derrière la voiture est partiellement hors champ à l'instant suivant. L'estimation de cette portion de sol est correcte quand l'algorithme a été entraîné à estimer les occultations en plus du flot alors qu'elle est erronée sinon.

TABLEAU 5.2 – Comparaison des résultats d'estimation de cartes d'occultations (score F_1), et du nombre de paramètres du modèle, en fonction du module d'occultations utilisé. Résultats obtenus sur l'ensemble d'entraînement de MPI Sintel. Expériences réalisées en biframe. Les meilleurs résultats sont en caractères gras.

Modèle	Occultations (score F_1)		Paramètres	
	Clean	Final	nbr.	relatif
PWC-Net	-	-	8.64M	0 %
PWC-Net + OccHur	65.9 %	62.0 %	16.19M	+87 %
PWC-Net + OccNeoral	65.1 %	61.5 %	9.68M	+12 %
PWC-Net + OccLate	**66.0 %**	**62.3 %**	**8.68M**	**+0.5 %**

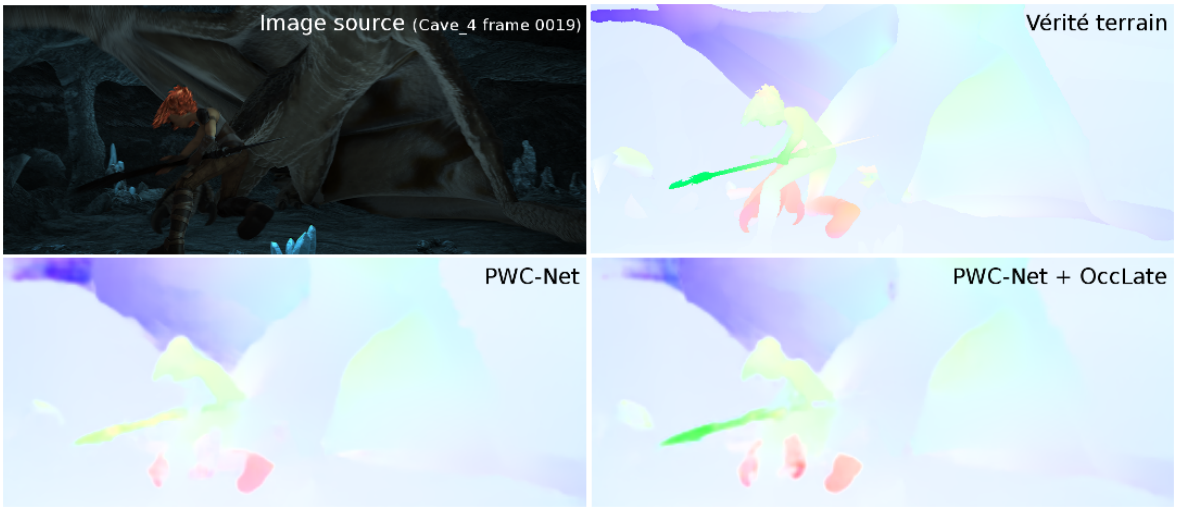

FIGURE 5.4 – Comparaison, pour la 19e paire d'images de la séquence "Cave_4" de Sintel Final, des flots optiques obtenus avec et sans module d'occultation. L'ajout d'un module d'estimation d'occultations améliore l'estimation de flot optique.

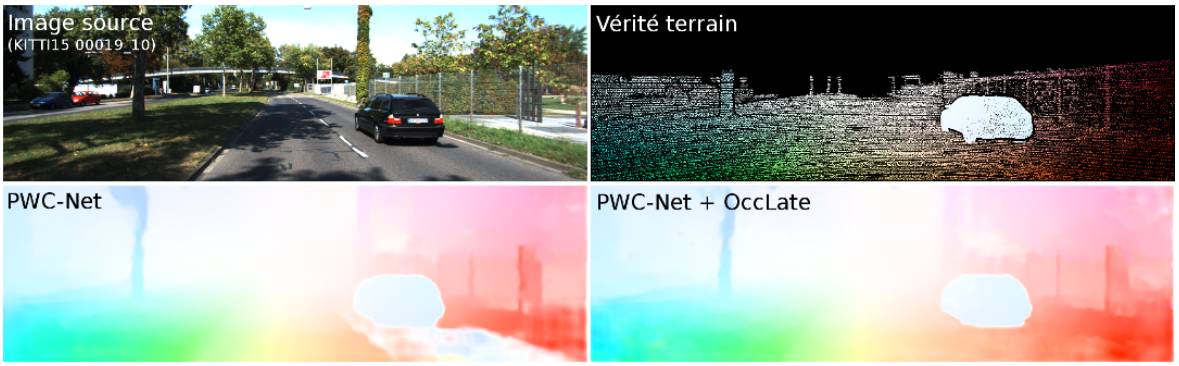

FIGURE 5.5 – Comparaison, pour la 20e séquence de l'ensemble d'entraînement de KITTI 2015, des flots optiques obtenus avec et sans module d'occultation. L'ajout d'un module d'estimation d'occultations améliore l'estimation de flot optique.

Comparons maintenant, entre elles, les trois variantes réalisant l'estimation conjointe. Du point de vue de l'EPE sur l'estimation du flot optique, tableau 5.1, les trois variantes présentent des résultats très proches sur Sintel ; et, sur KITTI, notre méthode donne de meilleurs résultats. Pour le score F_1 des cartes d'occultations, tableau 5.2, notre méthode "OccLate" donne des résultats équivalents à ceux de "OccHur", et meilleurs que ceux de "OccNeoral". Enfin, en comparant la taille des modèles en termes de nombre de paramètres, on remarque que notre méthode — tout en donnant des résultats aussi bons voire meilleurs — est très nettement plus légère que les deux autres.

Ces résultats confirment qu'en partageant la quasi-totalité des poids pour ces deux tâches très proches on obtient des représentations communes qui sont très bien adaptées à chacune des deux tâches. Ceci permet d'obtenir une architecture nettement plus légère et aussi performante, voire plus, que celles proposées par NEORAL et collab. [2018] et HUR et ROTH [2019]. Les trois méthodes présentées nécessitent cependant une supervision supplémentaire, et donc la disponibilité de la VT des cartes d'occultations. Il serait intéressant, à l'avenir, d'aller vers un mécanisme auto-supervisé pour apprendre à détecter les occultations, comme proposé par ZHAO et collab. [2020] dans leur méthode MaskFlownet.

5.3 Construction de l'architecture récurrente STaRFlow

5.3.1 Présentation globale

Le principe de notre architecture, schématisée sur la figure 5.6, est d'invoquer une même cellule récurrente, appelée *STaRCell*, à chaque instant et à chaque échelle. À chaque appel, la *STaRCell* réalise une estimation conjointe du flot optique et des cartes d'occultations, à partir :

— des descripteurs (appris) calculés par un encodeur (représenté en vert) pour la paire d'images de l'instant courant.

— de ses sorties précédentes, *ie* le flot et les occultations à l'échelle précédente. Il s'agit de la récurrence spatiale, représentée par des flèches grises.

— d'un état de mémoire, transmis d'un instant au suivant via une connexion temporelle, représentée en rouge.

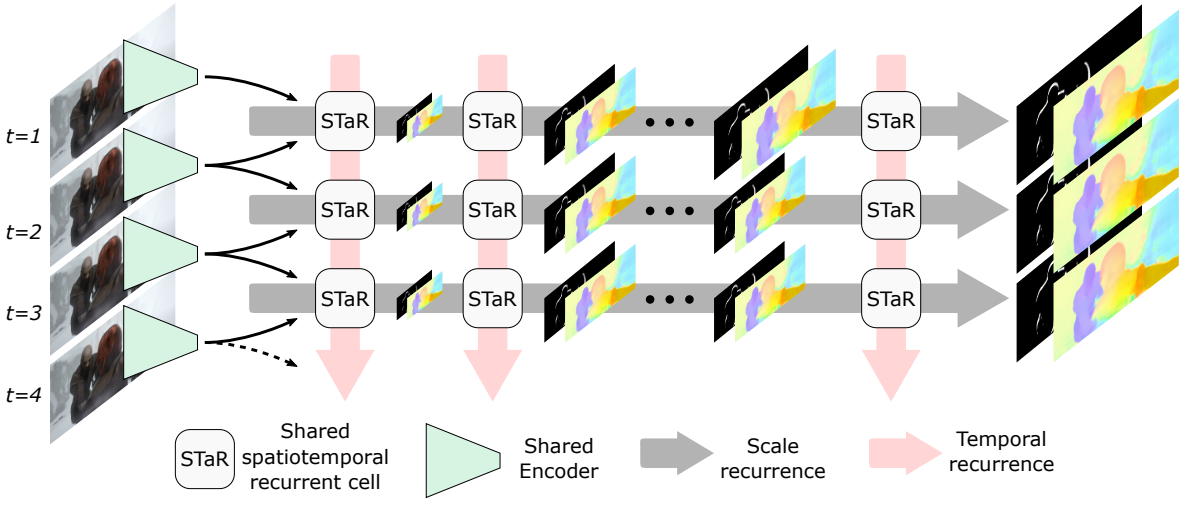

FIGURE 5.6 – Schéma de principe global de notre architecture récurrente.

La figure 5.6 montre une version dépliée de la récurrence spatio-temporelle sur 3 instants et 3 échelles.

L'encodeur, en vert sur la figure 5.6, est celui de l'architecture PWC-Net (SUN et collab. [2018b]). Il est le même (les paramètres sont partagés à la manière d'un réseau siamois) pour chacune des images de la séquence d'entrée, et produit pour chacune d'elles des descripteurs multi-échelles appris. La cellule récurrente, dont le fonctionnement est détaillé plus loin, prend en entrée les descripteurs de deux images consécutives, ainsi qu'une estimation grossière (de l'échelle précédente) du flot et des occultations. Elle donne en sortie une estimation affinée, à l'échelle courante, du flot et des occultations. Il s'agit là de la récurrence spatiale, symbolisée par les flèches horizontales grises sur la figure 5.6. À partir du deuxième instant de la séquence, la récurrence temporelle transmet des informations du passé, qui sont prises comme une entrée supplémentaire pour l'estimation à l'instant suivant, ceci est représenté par les flèches verticales rouges. Cette double récurrence permet d'utiliser les mêmes paramètres appris pour toutes les échelles et tous les instants, ceci permet de contenir la taille du modèle.

5.3.2 La cellule récurrente spatio-temporelle

Description des récurrences

La figure 5.7 montre le fonctionnement interne de la cellule récurrente, estimant le flot U_t^l et les occultations O_t^l à l'échelle l et à l'instant t. L'architecture de la cellule récurrente est inspirée de l'estimateur de flot de l'architecture PWC-Net (SUN et collab. [2018b]) : on en retrouve la corrélation des descripteurs de l'encodeur F_t^l et F_{t+1}^l, après application du flot U_t^{l-1}, estimé à l'échelle précédente, aux descripteurs F_{t+1}^l de l'image cible ; le *CNN optical flow estimator* (aussi appelé *decoder*) avec convolutions denses, et le *context network*. La cellule implémente également l'estimation d'occultations "OccLate" présentée en section 5.2.1, par simple ajout d'une sortie (et donc d'une entrée) et de connexions résiduelles supplémentaires (en bleu).

La récurrence spatiale tient au fait que l'estimation à tous les niveaux d'échelle utilise les mêmes opérations apprises. La STaRCell apprend donc l'opération "réaliser l'estimation résiduelle pour passer à l'échelle suivante" indépendamment de l'échelle. Il s'agit en fait du fonctionnement résiduel itératif proposé par HUR et ROTH [2019], qui nous permet de limiter notablement le nombre de poids nécessaires à la mise en place de notre récurrence temporelle, présentée ci-après.

Afin de réaliser une estimation multiframe récurrente, nous ajoutons une connexion temporelle, en rouge sur la figure 5.7, qui prélève des activations à un stade avancé (conceptuellement proche des sorties) de l'estimation à l'instant t et les ajoute aux entrées de l'estimateur à $t+1$. La structure de l'estimateur CNN de flot optique faisant intervenir des connexions denses, le nombre de canaux du tenseur prélevé proche de la sortie est bien supérieur au nombre de canaux en entrée de la connexion temporelle. Afin d'avoir un nombre de canaux constant en entrée de la connexion temporelle à chaque instant, et ainsi de pouvoir réaliser le bouclage, on réduit le nombre de canaux au moyen d'une convolution 1x1, qui calcule chacun de ses canaux de sortie comme une somme pondérée de ses canaux d'entrée[1]. Le tenseur transmis dans la connexion temporelle comporte 116 canaux après compression par la convolution 1x1.

Pour la première paire d'images d'une séquence, on ne dispose d'aucune information sur le passé. L'estimation pour ce premier instant est réalisée en biframe. En pratique

1. On peut voir cela comme une couche complètement connectée, mais dans la dimension des canaux.

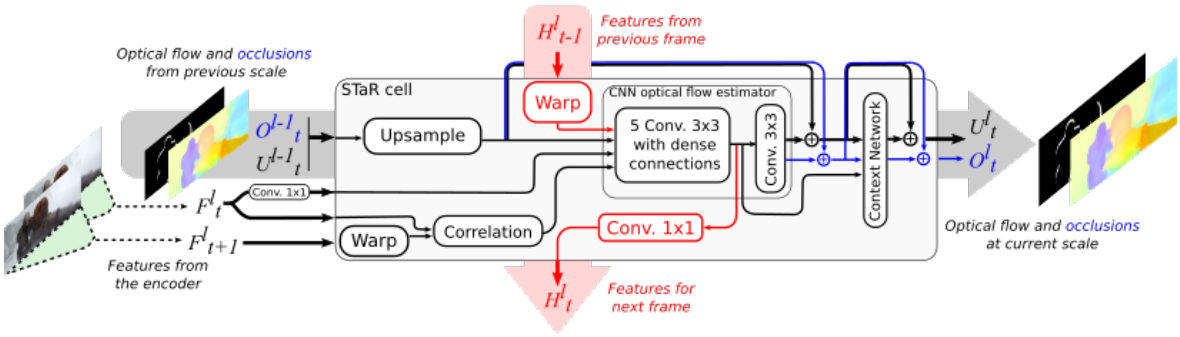

FIGURE 5.7 – Schéma structurel de la cellule récurrente en temps et en espace (STaRCell).

l'état de mémoire est initialisé avec des zéros.

Notre récurrence temporelle se distingue de celle de ContinualFlow (NEORAL et collab. [2018]). Là où la connexion temporelle de ContinualFlow transmet le flot optique estimé à l'instant précédent, la nôtre transmet un état de mémoire appris, sous la forme de cartes d'activations. Le contenu de cette mémoire est appris pendant l'entraînement et peut potentiellement représenter :

— les flots optiques d'instants précédents.

— des portions de trajectoires sur un horizon temporel plus ou moins long.

— des informations sur les occultations entre objets.

— des pondérations régissant l'importance à donner, pour l'estimation suivante, aux différentes informations retenues du passé.

La possibilité de retenir une information relative à plusieurs instants du passé, et pas uniquement $t-1$, peut donner accès à la notion d'accélération, qui peut s'avérer utile pour prédire l'évolution future du mouvement.

Recalage de l'état de mémoire (*warping*)

Dans une scène en mouvement, un même objet apparaîtra à des positions différentes sur des flots estimés à des instants différents. Dans le cas de ContinualFlow, le flot de l'instant précédent $t-1$ (aligné sur la géométrie de l'image I_{t-1}) doit donc être recalé avant d'être concaténé aux autres entrées qui sont alignées sur la géométrie de I_t. Dans notre cas, c'est l'ensemble des cartes d'activation prélevées par la connexion temporelle, constituant l'état de mémoire, qui doit être recalé dans la géométrie de l'instant suivant. Dans les deux cas il faut appliquer la transformation géométrique compensant le mouvement entre les instants $t-1$ et t.

Il est classique, dans les méthodes de flot optique, d'utiliser un *backward warp* pour rééchantillonner l'image I_{t+1} dans la géométrie de l'image I_t, en utilisant le flot optique $\boldsymbol{u}_{t\to t+1}$ de I_t vers I_{t+1}. On reconstruit ainsi une approximation de I_t en utilisant les intensités de I_{t+1} :

$$\hat{I}_t(\boldsymbol{k}) = \text{interp2}\left(I_{t+1}, \boldsymbol{k} + \boldsymbol{u}_{t\to t+1}(\boldsymbol{k})\right)$$

une interpolation bilinéaire est nécessaire pour obtenir les valeurs de l'intensité aux coordonnées spatiales non entières.

Pour utiliser ce *backward warp* dans le cas de la connexion temporelle, nous avons besoin du flot optique $\boldsymbol{u}_{t\to t-1}$ de I_t vers I_{t-1}, il s'agit du flot retour alors que notre algorithme calcule le flot direct, de I_{t-1} vers I_t. Cette méthode nécessite donc le calcul d'un flot supplémentaire à chaque instant. Ce flot retour est obtenu par inférence biframe du

même réseau, en inversant l'ordre de ces deux images. Le flot rééchantillonné dans la géométrie de I_t s'écrit alors :

$$\hat{\boldsymbol{u}}_{t-1\to t}(\boldsymbol{k}) = \text{interp2}\left(\boldsymbol{u}_{t-1\to t}, \boldsymbol{k} + \boldsymbol{u}_{t\to t-1}(\boldsymbol{k})\right) \tag{5.1}$$

Pour notre connexion temporelle nous utilisons cette même opération, appliquée aux cartes d'activation prélevées.

Évaluation de la récurrence temporelle

Nous présentons ici une série d'expériences pour évaluer notre récurrence temporelle. Pour une même architecture biframe de départ, nous comparons

— les résultats de l'estimation biframe, sans récurrence temporelle.

— les résultats, multiframe, obtenus en ajoutant à cette architecture la récurrence temporelle proposée dans ContinualFlow.

— les résultats, multiframe, obtenus en remplaçant cette récurrence temporelle par la nôtre.

Le code de ContinualFlow n'étant pas disponible au moment de nos travaux, nous proposons une implémentation "TRFlow" reprenant l'idée d'une connexion temporelle récurrente transmettant les flots estimés d'un instant au suivant. "TR" signifie "récurrence temporelle" (*Temporal Recurrence*). Nous appelons "TRFeat" notre connexion temporelle transmettant les cartes d'activations issues de l'instant précédent.

Cette comparaison est effectuée pour différentes architectures initiales :

— PWC-Net (SUN et collab. [2018b]), que nous réentraînons.

— PWC-Net + OccLate, avec notre estimation conjointe des cartes d'occultations.

— PWC-Net + OccLate + SR, "SR" représentant la récurrence spatiale (*Spatial Reccurence*).

À la manière de HUR et ROTH [2019], les architectures "SR" utilisent le même estimateur de flot, avec les mêmes poids appris, pour toutes les échelles. Les autres architectures utilisent des poids différents à toutes les échelles, comme SUN et collab. [2018b].

Les résultats présentés ci-après ne sont pas comparables à ceux de l'état de l'art, en particulier à ceux de ContinualFlow et IRR-PWC car les architectures que nous utilisons ici n'implémentent pas de module de *refinement*. D'autre part, aucun *finetuning* n'est réalisé après la phase d'entraînement sur FlyingThings. Donc l'évaluation de ces méthodes sur les ensembles de "*training*" de MPI Sintel et KITTI15 témoigne de leurs performances dans un cadre général, ces données étant bien différentes de celles vues lors de l'entraînement. On proposera plus tard dans ce chapitre, en section 5.5.1, une évaluation dans un contexte sur lequel l'algorithme est spécialisé, en réalisant un *finetuning* sur l'ensemble de *training* de Sintel (respectivement, de KITTI) et en évaluant sur l'ensemble de *test* de Sintel (respectivement, de KITTI).

Le tableau 5.3 rassemble les résultats de ces expériences. Remarquons tout d'abord que, quel que soit le choix de l'architecture de base, notre récurrence temporelle apprise (TRFeat) donne de meilleurs résultats que l'utilisation du flot de l'instant précédent (TRFlow), méthode décrite par NEORAL et collab. [2018]. On peut voir que ce gain est plus marqué sur les images dégradées, par comparaison entre Sintel Final et Sintel Clean, et sur les images réelles de KITTI 2015. Notons que sur KITTI 2015, la seule prise en compte du flot de l'instant précédent (TRFlow) n'améliore pas, voire dégrade, les résultats par rapport à la méthode biframe. Le gain, de notre méthode TRFeat par rapport au biframe

TABLEAU 5.3 – Comparaison des résultats de flot optique (erreur *endpoint* [px]), en fonction de la connexion temporelle utilisée, sur les ensembles d'entraînement de MPI Sintel et KITTI 2015. Les meilleurs résultats sont en caractères gras. Fl-all, sur KITTI, représente le pourcentage d'*outliers* (EPE > 3 px). *SR* représente la récurrence spatiale (*spatial recurrence*).

Méthode	Sintel Clean			Sintel Final			KITTI 2015		Paramètres	
	all	noc	occ	all	noc	occ	epe-all	Fl-all	nombre	relatif
Commun : PWC-Net										
Biframe	2.74	1.46	16.48	4.18	2.56	21.70	11.75	33.20 %	8.64M	0 %
TRFlow	2.47	**1.41**	13.97	4.01	2.52	20.00	11.27	33.77 %	8.68M	+0.5 %
TRFeat	**2.45**	1.44	**13.36**	**3.76**	**2.46**	**17.82**	**9.94**	**32.12 %**	12.31M	+42.5 %
Commun : PWC-Net + OccLate										
Biframe	2.46	1.32	14.82	3.96	2.47	20.06	10.58	31.28 %	8.68M	+0.5 %
TRFlow	2.17	1.23	12.33	3.90	2.50	19.11	10.82	32.51 %	8.73M	+1.0 %
TRFeat	**2.09**	**1.21**	**11.63**	**3.43**	**2.24**	**16.24**	**8.79**	**28.18 %**	12.38M	+43.3 %
Commun : PWC-Net + OccLate + SR										
Biframe	2.29	**1.20**	14.03	3.72	**2.32**	18.77	10.74	31.35 %	3.37M	−61.0 %
TRFlow	2.20	1.25	12.40	3.98	2.56	19.38	11.00	35.23 %	3.38M	−60.9 %
TRFeat	**2.10**	1.22	**11.67**	**3.49**	**2.32**	**16.15**	**9.26**	**30.75** %	4.37M	−49.4 %

et à TRFlow, apparaît également important dans les zones qui sont occultées dans l'image cible, voir la métrique "occ" dans les colonnes relatives à Sintel. L'estimation multiframe permet, en effet, de réaliser un "*inpainting* temporel" du flot optique dans les régions qui sont occultées à l'instant $t+1$ (image cible) mais visibles à t et aux instants précédents. L'estimation du mouvement réalisée précédemment peut être utilisée pour prédire le mouvement entre t et $t+1$, là où l'information issues des images I_t et I_{t+1} est insuffisante.

Les résultats qualitatifs des figures 5.8 à 5.10 permettent de mieux cerner les cas où notre récurrence temporelle apporte une amélioration notable.

— **Dans le cas de dégradations de la qualité image**. En comparant les résultats, pour une même séquence, sur les images de Sintel Clean et celles de Sintel Final, la figure 5.8 montre un cas où l'estimation multiframe est plus robuste aux dégradations de la qualité image que l'estimation biframe. Dans le cas des images "clean", toutes les méthodes donnent une bonne estimation, mais dans le cas "final" pour lequel la qualité image est dégradée, l'estimation biframe est également dégradée, tandis que la récurrence temporelle permet de conserver une estimation de bonne qualité.

— **Pour l'estimation du mouvement d'objets fins**. Comme le montre la figure 5.9, notre connexion temporelle apprise présente une meilleure sensibilité aux mouvements des objets fins (voir le poignard) comparé à la connexion temporelle TRFlow et à l'estimation biframe.

— **En présence d'occultations**. Dans le cas de la figure 5.10 l'estimation des occultations et la récurrence temporelle sont toutes les deux nécessaires pour réaliser une estimation correcte. L'estimateur biframe ne dispose pas de l'information nécessaire pour estimer le mouvement de la balise de signalisation visible en bas à droite du champ image, qui sera sortie du champ à l'instant suivant. L'estimateur multiframe pourrait en proposer une estimation approchée basée sur l'estimation

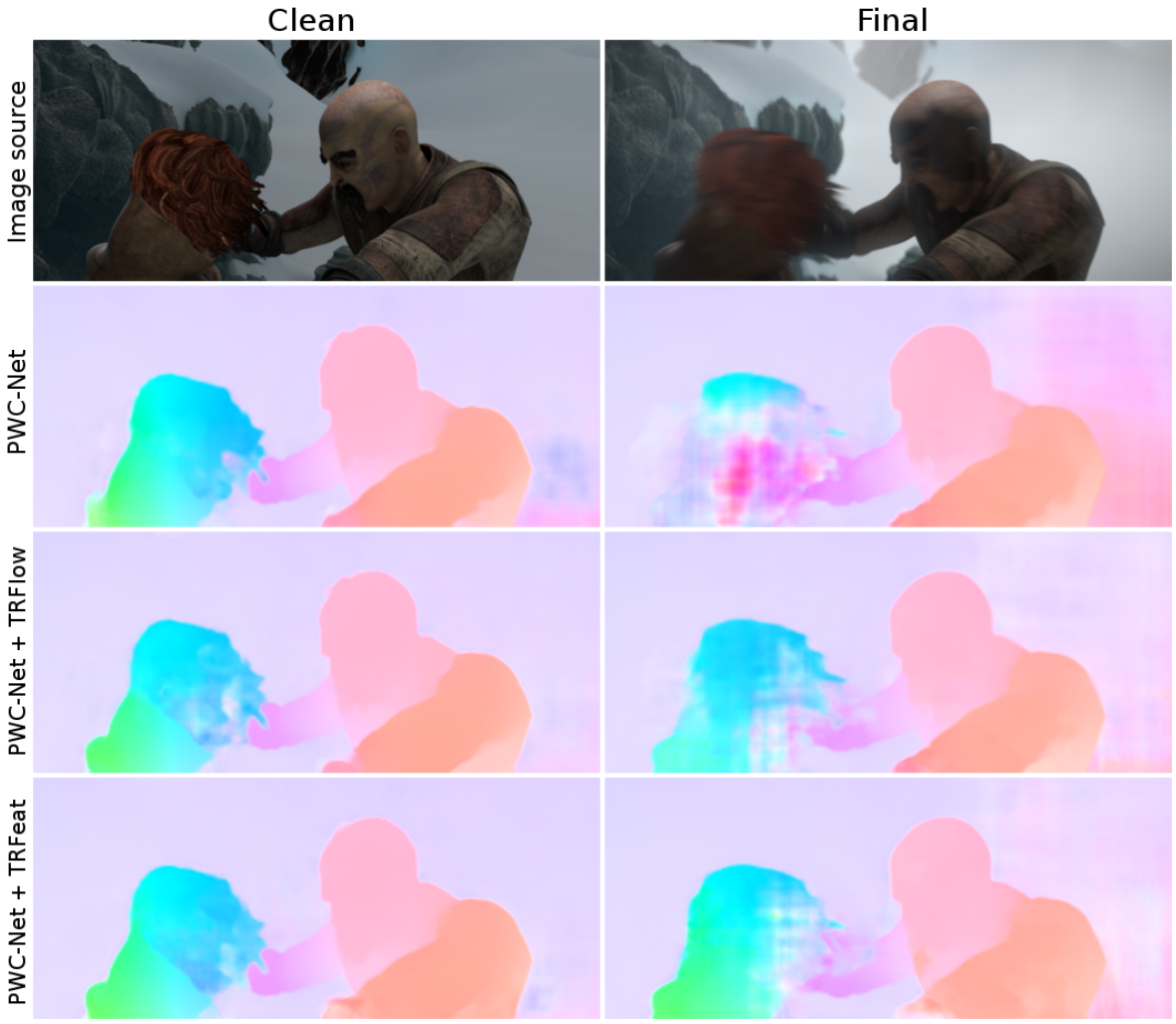

FIGURE 5.8 – Amélioration dans le cas d'une mauvaise qualité image, grâce à l'information temporelle, sur la 36e paire d'images de la séquence "Ambush_5" de Sintel. Pour les images "final", des dégradations simulées ont été ajoutée, comme du flou de bougé ou des effets atmosphériques.

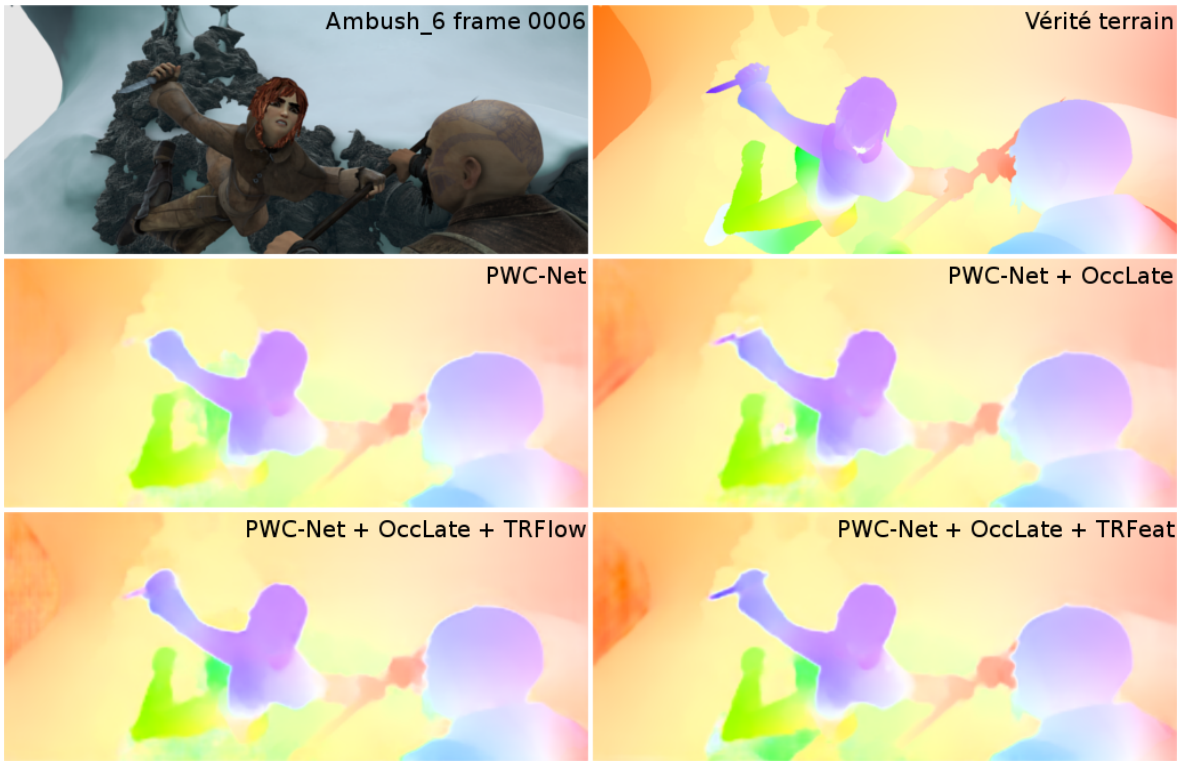

FIGURE 5.9 – Amélioration au niveau des objets fins sur la 6e paire d'images de la séquence "Ambush_6" de Sintel Clean. Le mouvement du couteau n'est détecté qu'avec notre méthode multiframe TRFeat. Ni la *baseline* biframe, ni la méthode TRFlow n'y parviennent.

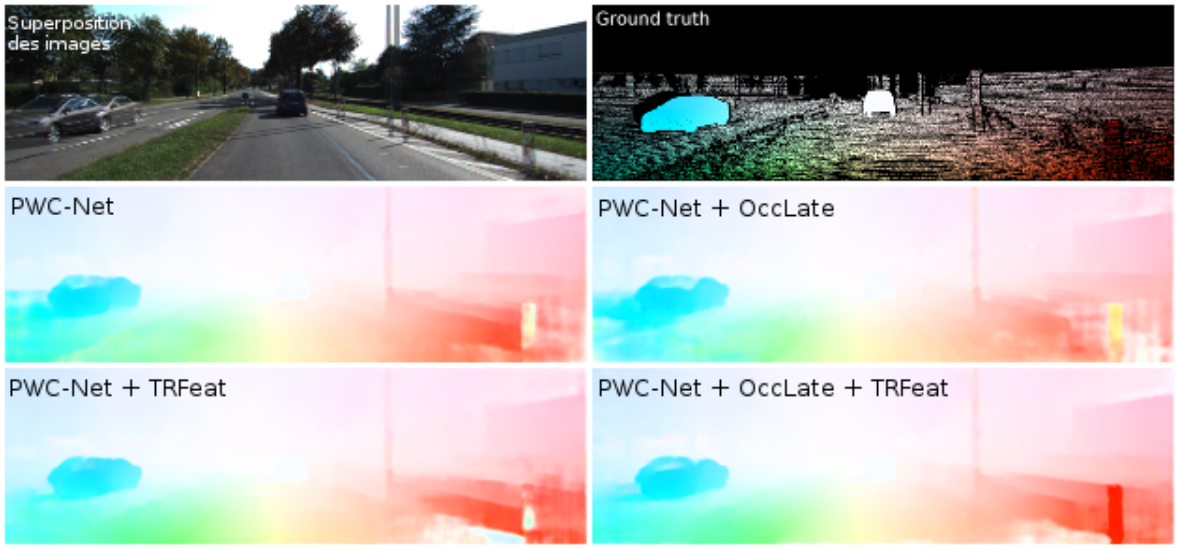

FIGURE 5.10 – Un cas dans lequel seule l'utilisation conjointe du multiframe et de l'estimation d'occultations permet d'estimer correctement le mouvement de la balise de signalisation en bas à droite de l'image. 106e séquence de l'ensemble d'entraînement de KITTI 2015.

TABLEAU 5.4 – Amélioration relative, en %, obtenue grâce à l'ajout de la tâche de détection des occultations, sur Sintel. Ces valeurs ont été calculées à partir de l'EPE moyenne "all" des sections "PWC-Net" et "PWC-Net + OccLate" du tableau 5.3.

	Clean	Final
Biframe	10.2	5.3
TRFlow	12.1	2.7
TRFeat	**14.7**	**8.8**

passée, mais n'y parvient que dans le cas où il a explicitement appris à détecter les occultations. L'apprentissage conjoint des tâches d'estimation du flot et des occultations rend donc l'estimation multiframe du flot plus robuste. Contraint d'apprendre à détecter les occultations avec les mêmes opérations, le réseau estime le mouvement en disposant de représentations lui permettant de mieux tirer profit de l'information temporelle.

Nous verrons en section 5.5.1 d'autres exemples, dans lesquels notre connexion temporelle permet

— d'améliorer l'estimation du mouvement d'objets fins (figure 5.15).

— de prédire le mouvement d'un objet occulté grâce à l'information du passé (figure 5.14).

Évaluation de la récurrence spatiale

Revenons au tableau 5.3, et comparons "PWC-Net + OccLate" avec et sans la récurrence spatiale (notée "SR"). Rappelons la différence entre les deux : avec "SR" les mêmes poids sont utilisés pour l'estimation à tous les niveaux d'échelle, tandis que sans "SR" on utilise des poids différents à chaque échelle. Dans le cas biframe, l'ajout de la récurrence spatiale améliore les résultats. Ceci correspond aux expériences de HUR et ROTH [2019], présentées dans leur "table 1", sur la comparaison de PWC-Net avec et sans l'option "IRR" et est cohérent avec les résultats qu'ils obtiennent. Dans le cas de notre méthode multiframe "TRFeat", cependant, on n'observe pas cette amélioration sur Sintel, et même une légère dégradation sur KITTI. L'ajout de la récurrence spatiale demeure particulièrement intéressant pour réduire la taille du modèle, comme le montre le relevé du nombre de paramètres associé à chaque méthode. Cette réduction est particulièrement importante dans le cas de notre méthode multiframe TRFeat, qui passe de 12.38 millions à 4.37 millions de paramètres, pour des performances pourtant équivalentes sur Sintel.

Amélioration apportée par l'estimation d'occultations

Enfin, pour compléter nos expériences précédentes sur l'amélioration de l'estimation du flot optique grâce à l'estimation conjointe des occultations, on remarque ici que les méthodes multiframe bénéficient également de cette amélioration. Plus précisément, on compare, dans le tableau 5.4, l'amélioration relative obtenue sur Sintel en ajoutant la tâche de détection d'occultations, pour les différentes architectures. L'amélioration obtenue est plus importante avec TRFeat. La prise en compte des occultations permet donc une meilleure exploitation de l'information temporelle, comme nous l'avons déjà remarqué précédemment (figure 5.10).

Influence du nombre d'images

Comme expliqué en section 5.4, les méthodes multiframe que nous comparons (TRFlow et TRFeat) sont entraînées avec des séquences de $N = 4$ images. Cela revient, pour notre méthode, à réinitialiser la mémoire temporelle toutes les 4 images. Rien n'empêche par la suite, pendant l'évaluation, de faire fonctionner ces deux méthodes avec des séquences de longueurs différentes. Dans le tableau 5.5 nous comparons leurs résultats pour l'estimation du dernier flot d'une séquence lorsque le nombre d'images N' consécutives de cette séquence augmente (avant réinitialisation de la mémoire). Les gains en performances, en augmentant N', sont plus importants pour TRFeat que pour TRFlow, et l'optimum de TRFeat est atteint pour un horizon temporel plus long que l'optimum de TRFlow. En propageant, d'un instant au suivant, un état de mémoire basé sur des cartes d'activations apprises, au lieu du flot estimé au dernier instant, notre méthode TRFeat permet d'exploiter des informations du passé à plus long terme que TRFlow. Ceci est plus marqué pour les images dégradées de Sintel Final, les images réelles de KITTI, ou encore dans les régions partiellement occultées (métrique "occ"). En outre, l'estimation de TRFeat continue de s'améliorer avec $N' = 5$. TRFeat semble donc capable d'apprendre à exploiter la continuité temporelle au-delà de la longueur des séquences présentées pendant l'entraînement.

La figure 5.11 présente les estimations obtenues en utilisant $N' = 3$ et $N' = 4$ (respectivement sur les 2^e et 3^e colonnes), pour TRFlow et TRFeat, et pour les 41^e et 42^e instants de la séquence Ambush_7 de Sintel Final. Pour l'instant 41, l'estimation est rendue compliquée par le fait que l'objet est très proche du bord du champ. TRFlow donne une mauvaise estimation dans le coin supérieur droit, que la séquence considérée comporte 3 ou 4 images. TRFeat donne une estimation correcte, même avec $N' = 3$. L'instant 42 est encore plus compliqué, d'une part car l'objet est encore plus proche du bord, d'autre part car celui-ci sera partiellement occulté à l'instant suivant. Pour cet instant, l'estimation dans le coin supérieur droit est erronée avec TRFlow pour $N' = 3$ et $N' = 4$, et avec TRFeat avec $N' = 3$. Avec une image supplémentaire, TRFeat parvient à résoudre le contour de l'objet. Notre connexion permet donc de profiter de la cohérence temporelle à plus long terme : pour la paire d'image "Ambush_7_0042", passer de 3 à 4 instants améliore l'estimation avec notre connexion mais pas avec celle proposée dans *ContinualFlow*.

5.3.3 Ajout du module de *refinement*

Les expériences présentées jusqu'ici montrent que notre mécanisme d'exploitation de la cohérence temporelle donne de meilleurs résultats que celui proposé par NEORAL et collab. [2018], et améliore nettement la qualité de l'estimation par rapport à l'architecture biframe sur laquelle on l'implémente. Pourtant à ce stade, notre méthode donne de moins bons résultats que ContinualFlow (NEORAL et collab. [2018]) et IRR-PWC (HUR et ROTH [2019]) sur les *benckmarks* Sintel et KITTI. Comme HUR et ROTH [2019], nous détectons les occultations, et proposons une récurrence spatiale en itérant sur les mêmes poids pour les différents niveaux d'un procédé multi-échelle. Cependant, contrairement à ContinualFlow et IRR-PWC, STaRFlow n'utilise pas de module de *refinement*. En ajoutant un tel module, reproduisant le fonctionnement de celui utilisé dans IRR-PWC, nous améliorons les résultats jusqu'à dépasser ContinualFlow et IRR-PWC sur les *benckmarks*.

Dans la suite, nous allons présenter notre méthode complète STaRFlow, avec *refinement*, puis comparerons les résultats obtenus avec et sans *refinement* pour montrer l'importance de cette étape sur la qualité finale du résultat. Notre méthode complète STaRFlow sera comparée aux autres méthodes de l'état de l'art dans la section 5.5.

TABLEAU 5.5 – Influence du nombre d'images N' utilisées au moment de l'évaluation. Les résultats sont donnés pour le flot estimé entre les instant $N'-1$ et N'. Les meilleurs résultats sont en caractères gras.

PWC-Net + OccLate + TRFlow + SR

	Sintel Clean			Sintel Final			Kitti15	
N'	all	noc	occ	all	noc	occ	epe-all	Fl-all
2	2.36	1.27	14.17	4.05	2.57	20.06	12.53	35.95 %
3	**2.17**	**1.24**	**12.29**	**3.95**	**2.56**	**19.03**	11.26	35.35 %
4	2.20	1.25	12.40	3.98	**2.56**	19.38	11.01	35.27 %
5	2.20	1.26	12.37	3.98	**2.56**	19.30	**10.94**	**35.17 %**
6	2.20	1.26	12.33	3.98	2.58	19.11	**10.94**	35.19 %

PWC-Net + OccLate + TRFeat + SR

	Sintel Clean			Sintel Final			Kitti15	
N'	all	noc	occ	all	noc	occ	epe-all	Fl-all
2	2.40	1.30	14.34	4.04	2.55	20.12	12.01	34.22 %
3	2.10	1.23	11.60	3.58	2.35	16.90	9.95	31.49 %
4	2.10	**1.22**	11.67	3.49	2.32	16.15	9.26	30.78 %
5	**2.08**	**1.22**	**11.36**	**3.43**	**2.27**	**15.99**	9.17	**30.66 %**
6	2.09	**1.22**	11.52	3.50	2.32	16.25	**9.14**	30.69 %

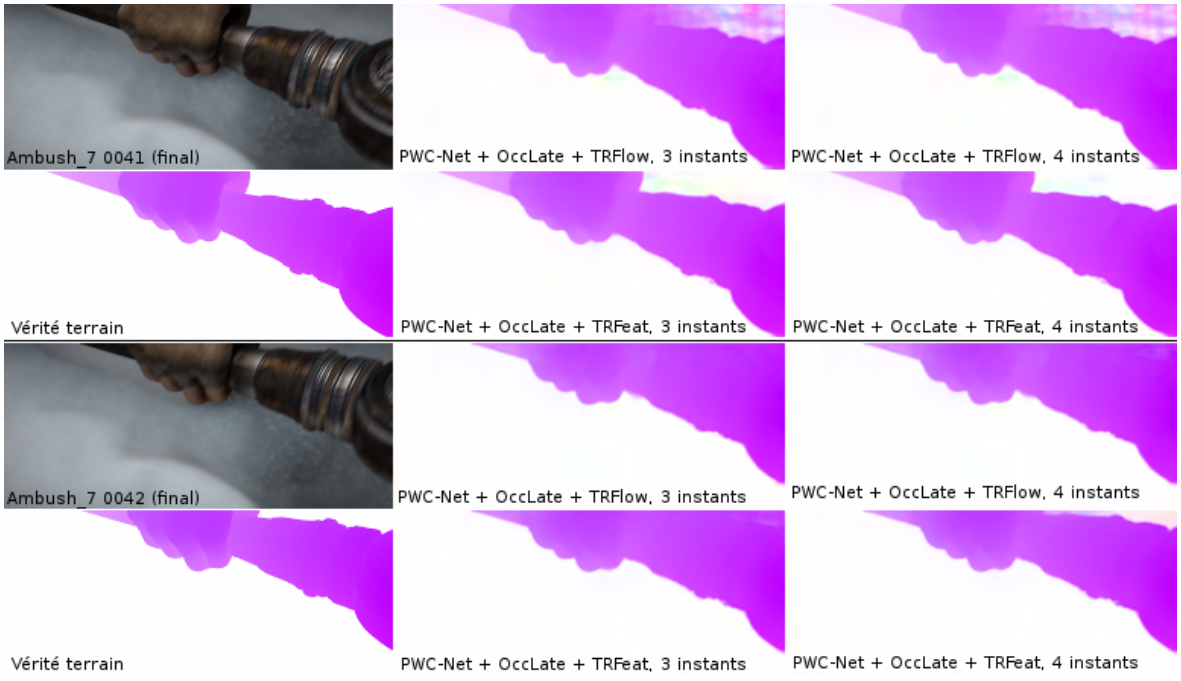

FIGURE 5.11 – Flots estimés en variant la longueur de l'horizon temporel considéré, pour les méthodes TRFlow et TRFeat. Avec notre méthode TRFeat, l'information temporelle peut être capturée sur un horizon temporel plus long, permettant de mieux détourer l'objet dans la partie en haut à droite du champ.

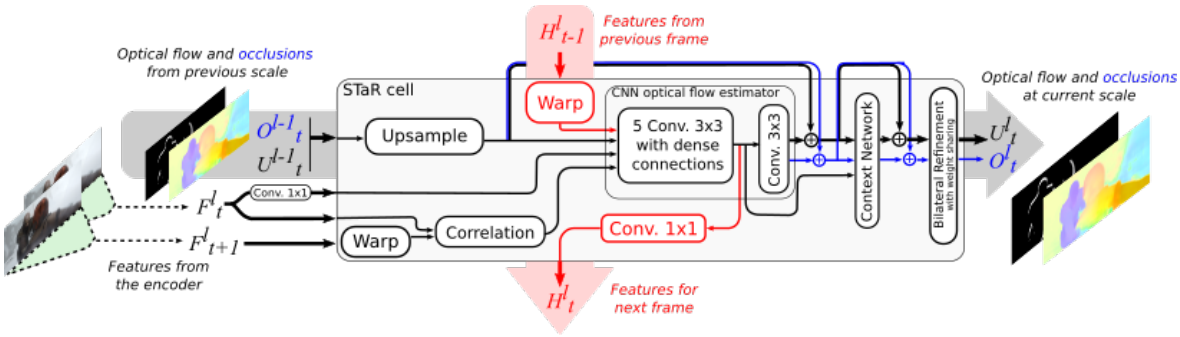

FIGURE 5.12 – Schéma structurel de la cellule récurrente en temps et en espace (*STaRCell*) avec le module de *refinement*.

TABLEAU 5.6 – Comparaison des résultats de STaRFlow avec et sans *refinement*. Erreur *endpoint* [px] sur le flot optique sur les ensembles d'entraînement de MPI Sintel et KITTI 2015. Les meilleurs résultats sont en caractères gras. Fl-all, sur KITTI, représente le pourcentage d'*outliers* (EPE > 3 px).

Méthode	Sintel Clean			Sintel Final			KITTI 2015		Nombre de paramètres
	all	noc	occ	all	noc	occ	epe-all	Fl-all	
Sans *refinement*	2.10	1.22	11.67	3.49	2.32	16.15	9.26	30.75 %	**4.37M**
Avec *refinement*	**1.63**	**0.88**	**9.79**	**3.08**	**1.97**	**14.99**	**6.76**	**22.47 %**	4.77M
Relatif (%)	−22.4	−27.9	−16.1	−11.7	−15.1	−7.2	−27.0	−26.9	+9.2

Description de la méthode

Nous ajoutons à STaRFlow le module de *refinement* proposé par HUR et ROTH [2019]. Le flot final, en sortie du module de *refinement*, est obtenu en calculant la convolution du flot à améliorer par des filtres spécifiquement générés pour chaque pixel du champ. Ces filtres sont générés par un CNN, entraîné avec le reste de l'architecture, qui prend en entrée les descripteurs issus de l'encodeur, le flot à améliorer et l'erreur photométrique (norme L2 de la différence entre l'image source et l'image cible pré-recalée). L'architecture de la nouvelle *STaRCell*, incluant le module de *refinement*, est représentée sur la figure 5.12. L'entraînement est effectué selon la méthode présentée en section 5.4.

Comparaison des résultats avec et sans *refinement*

Comme on peut le constater sur le tableau 5.6, l'ajout du module de *refinement* améliore globalement les résultats et de manière significative, pour une augmentation de la

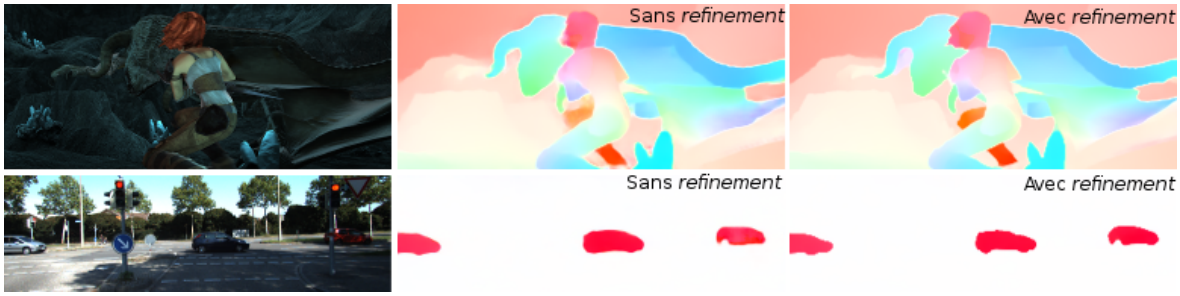

FIGURE 5.13 – Comparaison qualitative des résultats avec et sans *refinement*, sur un exemple de Sintel Final et un exemple de KITTI 2015.

taille du modèle de l'ordre de 10 %. Plus qualitativement, on peut voir sur la figure 5.13, que le *refinement* permet d'éviter certains effets de bavure des structures spatiales, et d'obtenir ainsi un résultat plus précis au niveau des discontinuités spatiales du champ de mouvement.

5.4 Entraînement d'une architecture récurrente pour le flot optique et les occultations

Le but de cette section est d'établir une procédure permettant d'entraîner un réseau de neurones à estimer le flot optique et les cartes d'occultations en tenant compte de plusieurs instants consécutifs [2], de manière récurrente. Dans un fonctionnement récurrent, le réseau estime le mouvement au fur et à mesure que les images défilent. Il voit d'abord deux images, et réalise une estimation biframe, puis il réalise une estimation pour chaque nouvelle paire d'images, en gardant une mémoire du passé.

5.4.1 La fonction de coût multiframe

Pour des séquences de N images, notre algorithme estime de manière récurrente les $N-1$ flots (et cartes d'occultations) des différentes paires d'images consécutives, dans l'ordre, en gardant une mémoire du passé pour les paires suivantes. La fonction de coût globale $\mathscr{L}$, permettant de superviser l'apprentissage, s'écrit comme une somme de $N-1$ fonctions de coûts pénalisant les estimations aux différents instants.

$$\mathscr{L} = \frac{1}{N}\sum_{t=1}^{N} \mathscr{L}_t \tag{5.2}$$

où $\mathscr{L}_t$ est, pour chaque instant t, la fonction de coût multi-échelle et multi-tâche suivante :

$$\mathscr{L}_t = \sum_{l=1}^{L} \alpha_l \left(\mathscr{L}_{\text{flow}}^{t,l} + \lambda \mathscr{L}_{\text{occ}}^{t,l} \right) \tag{5.3}$$

les coefficients α_l, relatifs à chacun des L niveaux d'échelle, sont ceux indiqués par Sun et collab. [2018b] : 0.32, 0.08, 0.02, 0.01 et 0.005, de l'échelle la moins résolue à la plus résolue. Le flot $u_t^l(x)$ à chaque instant t (*ie* entre I_t et I_{t+1}) et chaque échelle l est supervisé de la même manière que dans Sun et collab. [2018b] en utilisant la norme L_2 $\|\cdot\|$ sommée sur tous les pixels :

$$\mathscr{L}_{\text{flow}}^{t,l} = \sum \left\| u_t^l - u_{t,\text{GT}}^l \right\| \tag{5.4}$$

Pour la carte d'occultations o_t^l, la fonction de coût est l'entropie croisée pondérée pour deux classes (occulté, non occulté) :

$$\mathscr{L}_{\text{occ}}^{t,l} = -\frac{1}{2}\sum \left(w_t^l o_t^l \log o_{t,\text{GT}}^l + \bar{w}_t^l (1-o_t^l) \log(1-o_{t,\text{GT}}^l) \right) \tag{5.5}$$

sommée sur tous les pixels, et avec $w_t^l = \frac{H^l \cdot W^l}{\sum o_t^l + \sum o_{t,\text{GT}}^l}$ et $\bar{w}_t^l = \frac{H^l \cdot W^l}{\sum (1-o_t^l) + \sum (1-o_{t,\text{GT}}^l)}$, H^l et W^l étant les dimensions image à l'échelle l. Comme Hur et Roth [2019], nous calculons à chaque itération le coefficient λ qui équilibre les deux parties de la fonction de coût, de manière à pénaliser autant l'estimation du flot que celle des occultations.

2. L'annexe A donne plus de détails sur l'entraînement d'un CNN pour l'estimation de flot optique en général.

5.4.2 Jeux de données et hyper-paramètres

Depuis les travaux de ILG et collab. [2017], la méthode d'entraînement communément utilisée consiste à réaliser une première phase sur FlyingChairs et une seconde sur FlyingThings3D. FlyingChairs ne proposant que des paires d'images, nous allons, comme NEORAL et collab. [2018], entraîner en biframe lors de cette première phase (la connexion temporelle est inactive) et en multiframe seulement à partir de la deuxième phase, sur FlyingThings3D. Comme HUR et ROTH [2019], nous allons remplacer FlyingChairs par FlyingChairsOcc lors de la première phase, afin de superviser l'estimation des cartes d'occultation dès le départ. Il est courant d'effectuer des augmentations de données photométriques (ajout de bruit, correction gamma, entre autres) et géométriques (ajout de rotations, translations, mises à l'échelle, par exemple). Nous utilisons les mêmes augmentations de données que HUR et ROTH [2019], à une exception près : nous n'utilisons pas de transformations géométriques relatives (transformations différentes pour les deux images) lors de la phase multiframe. De telles transformations relatives modifient le champ de mouvement entre les deux images considérées. Cela n'est pas gênant dans le cas d'une estimation biframe, puisqu'il est possible de calculer le flot VT entre les nouvelles images dès lors qu'on connaît les transformations appliquées et la VT du flot avant application des transformations. En multiframe, cependant, le but est d'apprendre à profiter de la continuité temporelle, que des transformations relatives casseraient en modifiant l'évolution temporelle du mouvement.

Notre procédure d'entraînement consiste d'abord à pré-entraîner tout le réseau sauf la connexion temporelle (ce qui revient à retirer les flèches rouges sur les schémas des figures 5.6 et 5.7, en pratique des zéros sont transmis dans la connexion temporelle), sur les données biframe et 2D de FlyingChairsOcc. Nous utilisons des *mini-batchs* de 8 images et entraînons pendant 600 000 itérations, avec un taux d'apprentissage (*learning rate*) initial de 10^{-4}, qui est divisé par 2 toutes les 100 000 itérations après les 300 000 premières. Puis nous continuons l'entraînement sur des séquences de $N = 4$ images issues du jeu de données FlyingThings3D. À chaque début de séquence, la connexion temporelle est initialisée à zéro, puis à partir de la deuxième paire d'images elle contient des informations issues de l'estimation à l'instant précédent. Il est théoriquement possible d'utiliser des séquences plus longues, mais cela augmente l'espace mémoire requis pour l'entraînement, notamment à cause des calculs de gradients nécessaires à l'optimisation des poids du réseau. Pour cette seconde phase, nous utilisons des *mini-batchs* de 4 images et entraînons pendant 400 000 itérations. Comme la connexion temporelle est entraînée pour la première fois lors de cette phase, et comme suggéré par NEORAL et collab. [2018], le taux d'apprentissage initial pour cette phase est de 10^{-4}, soit significativement supérieur au taux de 10^{-5} utilisé pour l'entraînement des méthodes biframe sur FlyingThings3D (HUR et ROTH [2019]; ILG et collab. [2017]; SUN et collab. [2018a]). Ce taux d'apprentissage est divisé par 2 toutes les 100 000 itérations après les 150 000 premières. Les courbes d'évolution de la fonction de coût et de l'erreur de validation pour cet entraînement dans le cas de notre architecture STaRFlow sont données dans l'annexe A en section A.5.

Pour la phase multiframe, une séquence de $N = 4$ images est présentée au réseau de la manière suivante :

— on présente les deux premières images au réseau, qui effectue alors une estimation biframe (la connexion temporelle est inactive, remplie de valeurs nulles) et donne un flot et une carte d'occultations pour le premier instant.

— on présente ensuite les deuxième et troisième images au réseau, pour lequel la connexion temporelle est maintenant active, contenant des informations issues

de l'estimation à l'instant précédent. Le flot optique et la carte d'occultations du deuxième instant sont alors estimés, cette estimation tient compte de la paire d'images courante et d'une mémoire de l'instant précédent.

— on présente les images 3 et 4, la connexion temporelle véhiculant potentiellement des informations issues des estimations aux deux instants précédents, et l'estimation pour le troisième instant est réalisée.

— on pourrait continuer de la même manière pour $N > 4$ au prix d'une occupation mémoire plus importante pour le calcul des gradients nécessaire à l'optimisation des poids du réseau, tenant compte des $N-1$ estimations de précédentes.

— avant de présenter la séquence suivante, la mémoire est réinitialisée.

Les poids du réseau sont optimisés de manière à minimiser la fonction de coût décrite précédemment.

5.5 Comparaison à l'état de l'art

Nous reportons et commentons les résultats obtenus, pour notre méthodes STaRFlow et les autres méthodes de l'état de l'art, sur les *benchmarks* Sintel et KITTI. Ces résultats sont issus d'une évaluation sur les ensembles de "test" après spécialisation (phase d'entraînement supplémentaire dite de *finetuning*) du réseau sur les ensembles d'entraînement associés. Ensuite, nous comparons STaRFlow et IRR-PWC (la méthode biframe la plus proche) sans cette spécialisation, de sorte à évaluer leurs performances en généralisation à des données très différentes des données d'entraînement.

5.5.1 Résultats sur les ensembles d'évaluation de Sintel et KITTI

Jusqu'à présent nous avons arrêté l'entraînement après la phase sur FlyingThings3D, et réalisé l'évaluation sur les ensembles d'entraînement de Sintel et KITTI, pour lesquels la VT des flots optiques est publique. Nous allons maintenant comparer nos résultats aux autres méthodes de l'état de l'art sur les ensembles de test en soumettant nos résultats aux *benchmarks*. Pour cela, tout comme les méthodes concurrentes, nous effectuons un *finetuning* sur l'ensemble d'entraînement de Sintel (respectivement KITTI) avant l'évaluation sur l'ensemble de test de Sintel (respectivement KITTI). Pour cela nous procédons comme HUR et ROTH [2019] — qui reprennent le protocole proposé par SUN et collab. [2018a] pour entraîner PWC-Net+ — mais en considérant des séquences de $N = 4$ images au lieu des paires d'images. Il est possible avec le jeu d'entraînement de Sintel de superviser tous les instants. Cependant sur KITTI, un seul instant est annoté par séquence (instant central de séquences comportant 20 images). Dans ce cas nous choisissons nos séquences de 4 images de sorte à superviser le dernier instant, nous ne pouvons donc utiliser qu'une séquence de 4 images par séquence de 20 images disponible dans KITTI.

Le tableau 5.7 recense les résultats, en termes d'erreur *endpoint* moyenne, pour notre méthode STaRFlow et les principales méthodes concurrentes. Ces résultats sont disponibles sur les pages web des *benchmarks* MPI Sintel [3] et KITTI15 [4]. Au moment de la soumission, STaRFlow obtient le meilleur score sur Sintel Final parmi toutes les méthodes publiées.

Parmi les méthodes recensées dans le tableau — qui étaient les meilleures méthodes publiées au moment de nos travaux sur le sujet — STaRFlow est celle qui donne les

3. http://sintel.is.tue.mpg.de/results
4. http://www.cvlibs.net/datasets/kitti/eval_flow.php

TABLEAU 5.7 – Résultats sur les ensembles d'évaluation des *benchmarks* MPI Sintel and KITTI 2015. Erreur *endpoint* [px] pour Sintel, pourcentage d'*outliers* sur KITTI. Pour chaque colonne le meilleur résultat est en caractères gras, le suivant en italique. Les méthodes multiframe sont marquées *. †: valeur donnée par HUR et ROTH [2019], ‡: valeur donnée par LIU et collab. [2020].

Méthode	MPI Sintel clean	MPI Sintel final	KITTI 2015 Fl-all	Nombre de paramètres
ARFlow-mv*	4.49	5.67	11.79 %	**2.37M**
LiteFlowNet	4.54	5.38	9.38 %	5.37M
PWC-Net	4.39	5.04	9.60 %	8.75M
LiteFlowNet2	3.48	4.69	7.62 %	6.42M
PWC-Net+	3.45	4.60	7.72 %	8.75M†
IRR-PWC	3.84	4.58	7.65 %	6.36M
MFF*	3.42	4.57	7.17 %	9.95M
ContinualFlow_ROB*	3.34	4.53	10.03 %	14.6M†
SelFlow*	3.74	4.26	8.42 %	4.79M‡
MaskFlownet	**2.52**	4.17	**6.11 %**	N/A
ScopeFlow	3.59	*4.10*	*6.82* %	6.36M
STaRFlow*	*2.72*	**3.71**	7.65 %	*4.77M*

meilleurs résultats sur Sintel Final, est en deuxième position sur Sintel Clean et est à égalité avec IRR-PWC sur KITTI 2015.

Il est intéressant de comparer plus spécifiquement STaRFlow à ContinualFlow et IRR-PWC, qui sont des méthodes très proches (ContinualFlow est également une méthode multiframe récurrente, IRR-PWC est la méthode biframe la plus proche de STaRFlow). STaRFlow bat largement ContinualFlow sur Sintel Clean, Sintel Final et KITTI 2015, tout en ayant 3 fois moins de paramètres. La comparaison avec IRR-PWC montre clairement l'intérêt de l'estimation multiframe sur Sintel. Cependant, sur KITTI, IRR-PWC et STaRFlow sont à égalité : la cadence d'acquisition de 10 Hz des séquences de ce jeu de données est probablement insuffisante pour profiter de la même continuité temporelle que celle exploitée par STaRFlow sur Sintel; il est également possible que STaRFlow souffre d'un manque de supervision lors du *finetuning* sur KITTI puisqu'on ne dispose de la VT que pour un instant par séquence. La stratégie multiframe de MFF (REN et collab. [2019]), consistant à fusionner l'instant courant avec l'instant précédent recalé dans la géométrie courante, donne de meilleurs résultats sur KITTI.

On peut comparer plus qualitativement les résultats de notre méthode STaRFlow à ceux des méthodes ContinualFlow et IRR-PWC sur les exemples présentés sur les figures 5.14 et 5.15. Le mouvement du genou droit du personnage sur la figure 5.14 est perdu avec la méthode biframe IRR-PWC, tandis que les méthodes multiframe parviennent à estimer son mouvement. Ce genou est occulté dans l'image cible, il s'agit donc d'un cas d'*inpainting* temporel d'une zone d'occultations, comme expliqué plus tôt. Le résultat de STaRFlow présente une meilleure segmentation des objets, par rapport aux deux autres méthodes, sur les cheveux du personnage et sur les cornes et l'aile du dragon. Dans l'exemple de la figure 5.15, les doigts, fins, du personnage de droite sont nettement mieux segmentés avec STaRFlow.

Du point de vue du nombre de paramètres, seule la méthode ARFlow (LIU et collab. [2020]) est plus légère que STaRFlow. Les méthodes les plus légères après STaRFlow sont SelFlow (LIU et collab. [2019]) et LiteFlowNet (HUI et collab. [2018]). STaRFlow présente une précision bien meilleure que toutes ces méthodes.

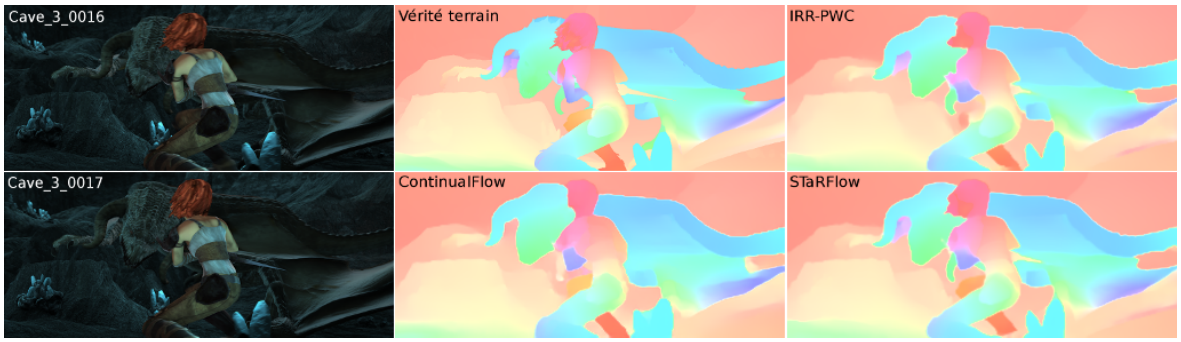

FIGURE 5.14 – Comparaison des flots estimés par IRR-PWC, ContinualFlow et STaRFlow, entre les 16e et 17e images de la séquence Cave_3 de Sintel Test Final. Résultats obtenus sur le site web du *benchmark*.

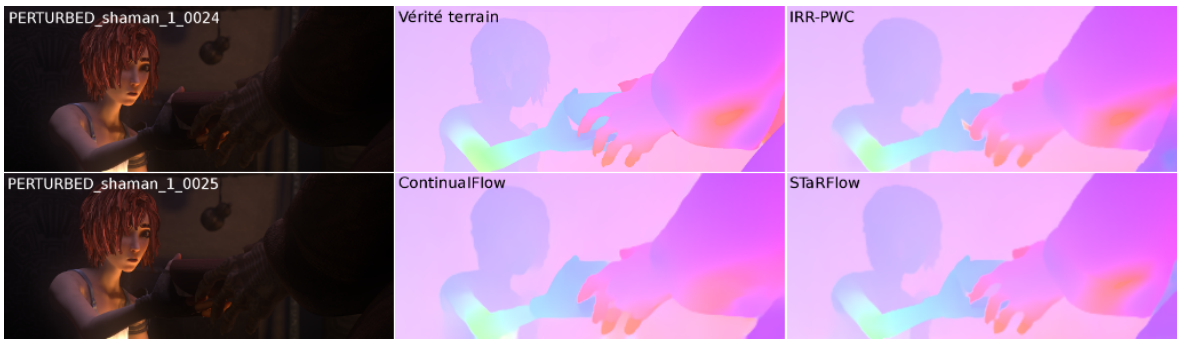

FIGURE 5.15 – Comparaison des flots estimés par IRR-PWC, ContinualFlow et STaRFlow, entre les 24e et 25e images de la séquence PERTURBED_shaman_1 de Sintel Test Final. Résultats obtenus sur le site web du *benchmark*.

5.5.2 Taille du modèle et temps de calcul

La comparaison du nombre de paramètres de STaRFlow à ceux des autres méthodes concurrentes, dans le tableau 5.7, montre la légèreté de notre méthode par rapport aux méthodes de l'état de l'art. Celle-ci tient au partage des poids du réseau pour :

— les tâches d'estimation de flot optique et de détection des occultations.

— les différents niveaux de l'estimation multi-échelle et les différents instants de l'estimation multi-temporelle.

Sur les images de Sintel (1024×436) le temps d'inférence de STaRFlow est de 0.22 secondes par paire d'images, sur un processeur graphique ou *graphics processing unit* (GPU) de gamme moyenne NVIDIA GTX 1070. Ce temps est encore un peu long pour un scénario temps réel, mais déjà bien inférieur à celui des méthodes variationnelles récentes comme EpicFlow (REVAUD et collab. [2015]) qui nécessitent plusieurs secondes pour chaque paire d'images.

5.5.3 Comparaison du point de vue de la généralisation

Les auteurs de la méthode IRR-PWC mettent à disposition leur code et les poids du modèle après entraînement complet (incluant le *finetuning* sur Sintel ou KITTI) et ceux avant le *finetuning*, correspondant donc à un entraînement uniquement sur FlyingChairs puis FlyingThings3D. Nous proposons de pousser la comparaison entre STaRFlow et IRR-PWC, sans *finetuning*. Ainsi, en évaluant les méthodes sur l'ensemble d'entraînement de Sintel, on les évalue dans un contexte qui leur est inconnu, en étant assez différent des images de FlyingChairs et FlyingThings, vues pendant l'entraînement. Cette évaluation nous paraît plus générale que celle d'une évaluation sur l'ensemble d'évaluation de Sintel après *finetuning* sur son ensemble d'entraînement, pour laquelle le réseau "connaît" déjà le contexte applicatif, puisqu'il a été entraîné sur des images semblables. Par ailleurs, IRR-PWC est une architecture biframe très proche de STaRFlow. Les seules différences sont :

— IRR-PWC a un décodeur dédié aux occultations ("OccHur" en section 5.2.2), tandis que dans STaRFlow nous séparons les tâches de flot et d'occultations au niveau de la dernière couche du réseau ("OccLate" en section 5.2.2).

— STaRFlow réalise une estimation multiframe, IRR-PWC est une méthode biframe.

— IRR-PWC utilise une fonction de coût pénalisant l'estimation du flot optique dans les deux sens (en inversant l'ordre des images), tandis que nous ne pénalisons que le flot direct (mais à plusieurs instants consécutifs).

La comparaison entre ces deux méthodes peut donc témoigner de l'apport d'un mécanisme multiframe récurrent.

Le tableau 5.8 présente la comparaison, en termes d'erreur sur l'estimation de flot optique, des méthodes IRR-PWC et STaRFlow sur les ensembles d'entraînement de MPI Sintel et KITTI 2015. La méthode STaRFlow est systématiquement meilleure, et plus significativement dans les zones occultées de Sintel et sur KITTI, tout en étant 25 % plus légère en termes de nombre de paramètres appris. Cette réduction de la taille du modèle est essentiellement due à notre implémentation "OccLate", partageant la majorité des poids pour les tâches de flot et d'occultations, tandis que IRR-PWC implémente un décodeur distinct pour les occultations. Notre connexion temporelle ajoute des paramètres mais, grâce à la récurrence spatiale ceux-ci sont partagés pour toutes les échelles, ce qui permet de limiter le nombre de paramètres nécessaire.

TABLEAU 5.8 – Comparaison des résultats de STaRFlow et IRR-PWC après entraînement sur FlyingChairs puis FlyingThings, sans *finetuning*. Erreur *endpoint* [px] sur le flot optique sur les ensembles d'entraînement de MPI Sintel et KITTI 2015. Les meilleurs résultats sont en caractères gras. Fl-all, sur KITTI, représente le pourcentage d'*outliers* (EPE > 3 px).

Méthode	Sintel Clean			Sintel Final			KITTI 2015		Nombre de
	all	noc	occ	all	noc	occ	epe-all	Fl-all	paramètres
IRR-PWC	1.86	0.92	13.86	3.43	2.11	20.21	9.08	26.80 %	6.36M
STaRFlow	**1.63**	**0.88**	**9.79**	**3.08**	**1.97**	**14.99**	**6.76**	**22.47 %**	**4.77M**
Relatif (%)	−12.4	−4.3	−29.4	−10.2	−6.6	−25.8	−25.6	−16.2	−25

Dans la comparaison des méthodes sur les *benchmarks*, tableau 5.7, IRR-PWC et STaRFlow étaient à égalité sur KITTI, et nous avions conclu que les mouvements y étaient trop violents pour pouvoir exploiter correctement la cohérence temporelle. Ces résultats correspondaient au cas de l'évaluation sur l'ensemble d'évaluation de KITTI après *finetuning* sur son ensemble d'entraînement, donc à un cas où le réseau s'est spécialisé sur des données de navigation routière — et issue des mêmes conditions d'acquisitions que celles utilisées pour l'évaluation. Les résultats du tableau 5.8 montrent au contraire une nette supériorité de STaRFlow. Ces résultats correspondent au cas d'une évaluation sur l'ensemble d'entraînement de KITTI après entraînement uniquement sur FlyingChairs et FlyingThings3D, donc à un cas où le contexte de la navigation routière est inconnu. STaRFlow présente donc, ici, une meilleure capacité de généralisation à des données inconnues. Revenons, donc, sur l'interprétation des résultats après *finetuning*, présentés dans le tableau 5.7. Il est probable que le *finetuning* sur KITTI conditionne le réseau à des mouvements très contraints, avec un mouvement global divergent (dû au mouvement propre de la caméra qui avance), et des objets mobiles qui suivent les routes et ont donc des trajectoires assez prévisibles même sans connaissance du passé, donc même pour une méthode biframe.

Comparons maintenant plus qualitativement les résultats de STaRFlow et IRR-PWC. Sur l'exemple de la figure 5.16, l'information temporelle permet à STaRFlow d'estimer correctement le mouvement de la main, dont l'estimation biframe est erronée (probablement à cause d'un changement d'orientation de la main, changeant son aspect visuel). Sur la figure 5.17, IRR-PWC échoue dans l'estimation de la jambe gauche du personnage, qui est partiellement occultée dans l'image cible, STaRFlow parvient à en donner une estimation correcte en exploitant la continuité temporelle. Le bras droit du personnage et le haut du dragon, en particulier une de ses cornes apparaissant à droite de la tête du personnage, sont également mieux estimés avec STaRFlow. La figure 5.18 montre l'exemple d'un objet fin mobile, raté par IRR-PWC mais détecté par STaRFlow. Dans cet exemple, un bambou, au centre de l'image, commence à tomber. Celui-ci se met en mouvement à l'instant précédent de celui qui est présenté, à ce moment là, STaRFlow ne le remarque pas non plus. Ce n'est qu'à la deuxième paire d'image pour laquelle le bambou est en mouvement, correspondant aux flots présentés sur la figure 5.18, que STaRFlow le détecte.

Pour terminer, la figure 5.19 montre une limite de STaRFlow. Dans cet exemple l'estimation biframe de la méthode IRR-PWC est bonne, tandis que STaRFlow introduit une erreur dans l'estimation du mouvement de la tête du personnage de gauche. Cet exemple correspond à un moment de la séquence où il y a un changement brusque de direction du mouvement du personnage de gauche, donc une discontinuité temporelle. Aux instants précédents, le déplacement allait en direction du coin inférieur droit, et à partir de l'ins-

FIGURE 5.16 – Comparaison, pour la 35e paire d'images de la séquence "Temple_3" de Sintel Clean, des flots optiques obtenus IRR-PWC et STaRFlow, entraînés uniquement sur FlyingChairs et FlyingThings3D.

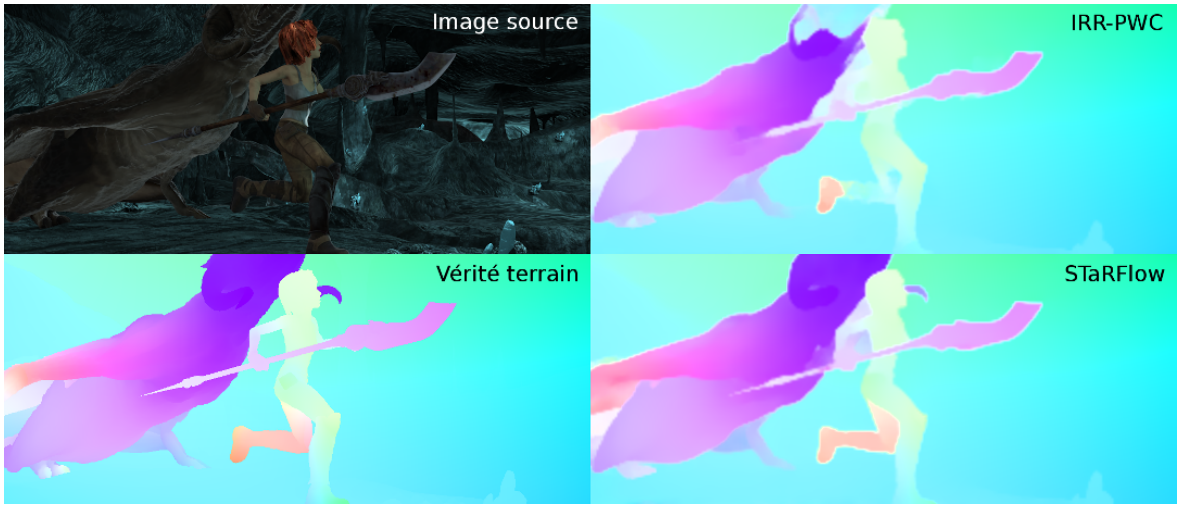

FIGURE 5.17 – Comparaison, pour la 17e paire d'images de la séquence "Cave_2" de Sintel Clean, des flots optiques obtenus IRR-PWC et STaRFlow, entraînés uniquement sur FlyingChairs et FlyingThings3D.

tant présenté sur la figure 5.19 la direction change pour aller vers la gauche. Au moment du changement, STaRFlow se trompe, en donnant trop d'importance à l'information du passé, et donne une estimation partant du principe que la tête continue de bouger dans la même direction. Cet exemple met en évidence une limite de STaRFlow, celle des ruptures de la continuité temporelle.

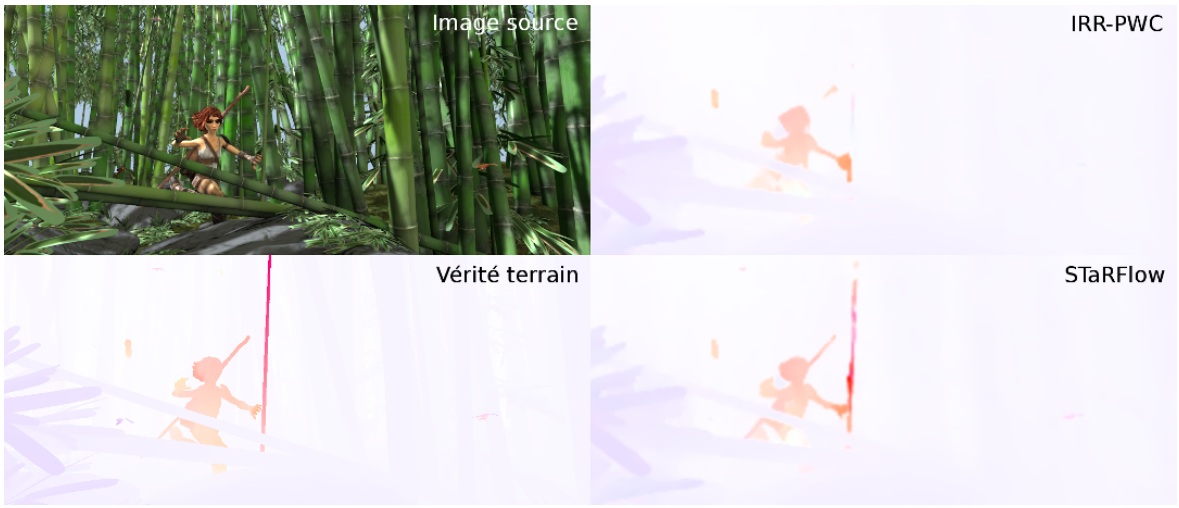

FIGURE 5.18 – Comparaison, pour la 25e paire d'images de la séquence "Bamboo_2" de Sintel Clean, des flots optiques obtenus IRR-PWC et STaRFlow, entraînés uniquement sur FlyingChairs et FlyingThings3D.

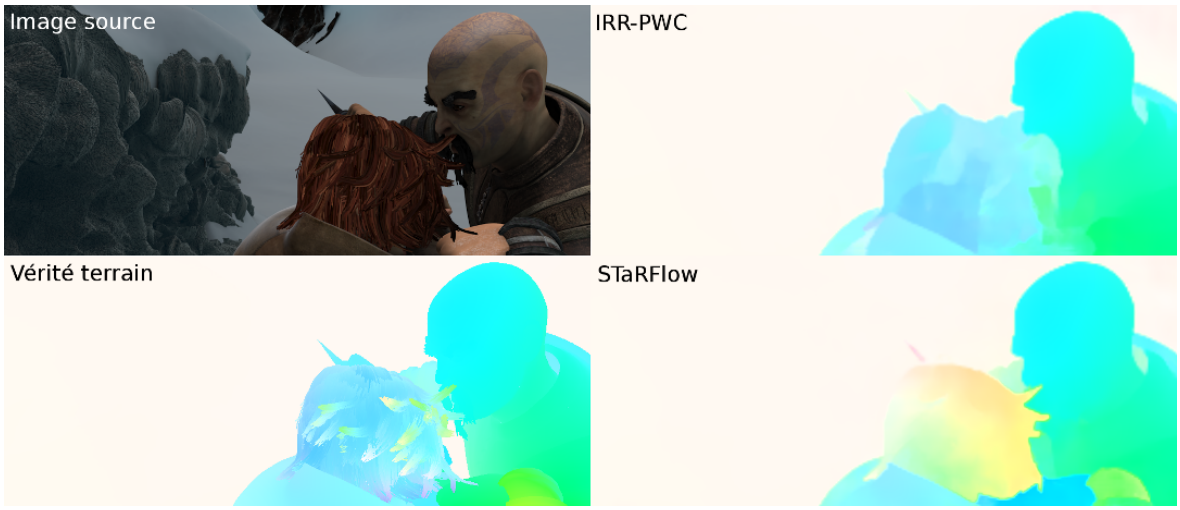

FIGURE 5.19 – Comparaison, pour la 46e paire d'images de la séquence "Ambush_5" de Sintel Clean, des flots optiques obtenus IRR-PWC et STaRFlow, entraînés uniquement sur FlyingChairs et FlyingThings3D.

5.6 Gestion des discontinuités temporelles

Nous avons montré, à la fin de la section précédente, un exemple où STaRFlow se trompe à cause d'un changement brusque de direction d'un objet. Il serait souhaitable qu'une méthode comme STaRFlow, en plus de bénéficier de la cohérence temporelle quand cela est possible, soit capable de détecter les ruptures de continuité temporelle et d'ignorer l'information du passé aux endroits et instants concernés par ces ruptures. Pour cela nous proposons d'agir à deux niveaux de notre méthode : sur l'architecture et sur les données d'entraînement.

5.6.1 Des données d'entraînement avec discontinuités temporelles

Afin de s'assurer de présenter au réseau des cas où il doit apprendre à ignorer une partie de l'information temporelle, nous incluons artificiellement des cas de rupture de la continuité temporelle dans les séquences d'entraînement. Pour cela nous modifions certaines des séquences de $N = 4$ images issues de FlyingThings3D avant de les présenter au réseau.

— pour un quart des séquences, arrivé à la troisième image on inverse le sens du temps pour montrer à nouveau la deuxième image (à la place de la quatrième).

— pour un autre quart des séquences, on montre les deux premières images, puis on recommence. Il y a donc deux inversions du temps. Autrement dit, on montre la première image, puis la deuxième, à nouveau la première et à nouveau la deuxième.

— la moitié restante des séquences est laissée intacte.

Cette méthode permet de créer des ruptures temporelles sans générer un nouveau jeu de données, mais ne présente que des cas de rupture où l'intégralité du champ spatial change de direction. Ceci peut représenter un biais important.

5.6.2 Un mécanisme de gestion de l'oubli dans l'architecture

Il existe des cellules récurrentes, initialement utilisées en traitement du langage puis adaptées à l'image pour réaliser des tâches de prédiction (BALLAS et collab. [2015]; CHERRIER et collab. [2017]; XINGJIAN et collab. [2015]), qui implémentent un mécanisme de portes afin de gérer la mise à jour d'un état de mémoire et la réalisation d'une prédiction à partir de cet état de mémoire et des entrées de l'instant courant. Il s'agit des *Long Short-Term Memory* (LSTM) et des *Gated Recurrent Unit* (GRU). Dans la suite nous allons décrire plus en détails les GRU, qui sont un peu plus simples.

Un GRU, comme illustré sur la figure 5.20, utilise deux portes afin de gérer l'information temporelle :

— la *reset gate* $r[t]$ permet de forcer l'oubli d'informations du passé.

— l'*update gate* $z[t]$ gère le mélange (somme pondérée) entre le passé et les nouvelles informations.

Ces "portes" sont implémentées en estimant, pour chacune, un masque multiplicatif grâce à une couche convolutive, et en multipliant pixel à pixel les entrées ou l'état interne par ce masque. Par application de la fonction *sigmoid*, ces masques prennent des valeurs comprises entre 0 et 1. Les couches convolutives qui estiment les masques prennent en entrée la concaténation de l'état interne $h[t-1]$ (représentant le passé) et des nouvelles entrées $x[t]$.

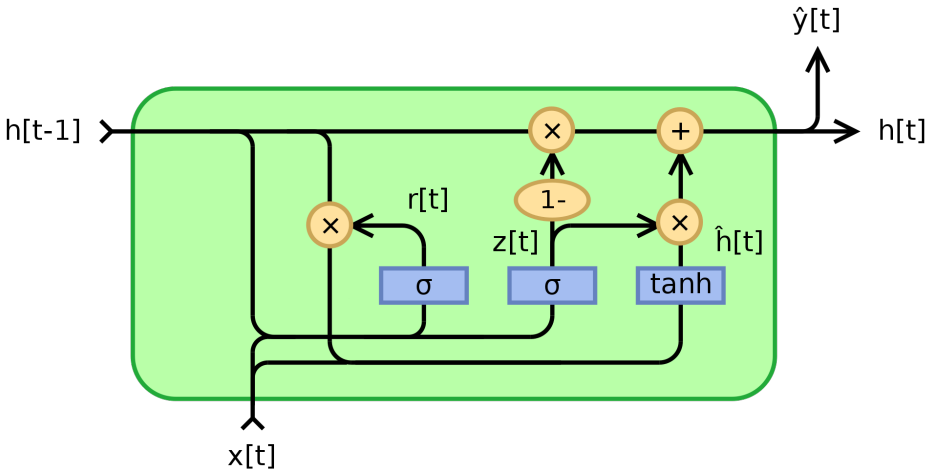

FIGURE 5.20 – Schéma d'un GRU. $h[t-1]$ est l'état retenu du passé, $x[t]$ représente les entrées courantes, $\hat{y}[t]$ est à la fois la sortie et l'état qui sera passé à l'instant suivant. Les opérations + et × sont réalisées pixel à pixel. Les rectangles bleus représentent des couches de convolutions (dans le cas d'un GRU convolutif, comme celui que nous utilisons) suivies d'une fonction *sigmoid* ou *tanh*. Image issue de Wikipedia, par Jeblad - Own work, CC BY-SA 4.0, https ://commons.wikimedia.org/w/index.php ?curid=66225938.

Nous proposons plusieurs nouvelles variantes de STaRFlow intégrant un mécanisme de gestion de l'oubli soit en incluant un GRU, soit en s'inspirant de son fonctionnement.

GatedSTaRFlow

GatedSTaRFlow est une variante de STaRFlow qui s'inspire de la *reset gate* du GRU pour implémenter un mécanisme d'oubli sélectif dans la connexion temporelle. Nous ajoutons une couche convolutive suivie d'un fonction *sigmoid*, estimant une porte multiplicative de même dimension que l'état de mémoire, à partir des entrées suivantes :

— l'état de mémoire issu de la connexion temporelle.

— les descripteurs donnés par l'encodeur pour l'image source.

— le résultat de la corrélation.

— le flot et les occultations de l'échelle précédente.

— le flot et les occultations pour la paire d'images de l'instant précédent, dans le sens inverse, *ie* de t vers $t-1$.

Par rapport à la STaRCell déjà présentée, nous avons ajouté le flot et les occultations de t vers $t-1$. La carte d'occultations en question donne les points visibles à t mais qui n'étaient pas visibles à $t-1$, ce sont donc des points pour lesquels l'information du passé est à manipuler avec précaution, car elle ne les concerne pas directement (néanmoins, l'information du passé peut concerner des points appartenant à un même objet et aider tout de même). Le flot de t vers $t-1$ informe sur l'origine des points, donc sur la direction de leur mouvement à l'instant précédent.

Juste avant d'être concaténé aux autres entrées de l'estimateur de flot et d'occultations (soit juste après le "warp" en rouge sur la figure 5.12), l'état de mémoire est multiplié par le masque obtenu.

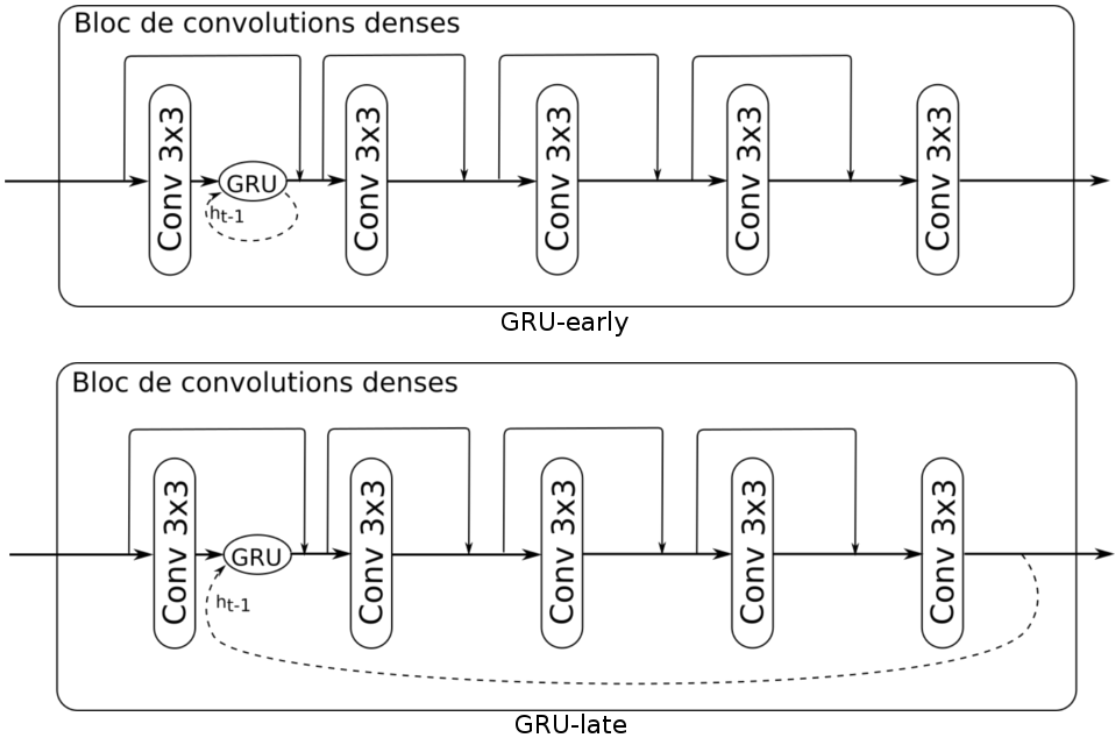

FIGURE 5.21 – Deux implémentations avec ajout d'un GRU dans le bloc de convolutions denses de STaRFlow.

Remplacer la connexion temporelle par un GRU

On peut retirer la connexion temporelle et ajouter un GRU dans l'estimateur de flot et d'occultations de STaRFlow. Ce GRU remplace la connexion temporelle en transmettant l'information temporelle via sa mémoire interne. Plus on place le GRU tôt dans l'enchaînement des couches de convolutions, plus le réseau peut en tenir compte pour réaliser son estimation. Si on place le GRU trop tard dans l'enchaînement, l'essentiel du processus d'estimation sera effectué en ne tenant compte que des entrées courantes. Cependant, il peut aussi être intéressant que le GRU travaille avec des représentations de haut niveau (*ie* proche de la sortie). Nous proposons de placer un GRU à l'issu de la première des 5 convolutions de l'estimateur, et de comparer deux variantes, "GRU-early" et "GRU-late", pour le bouclage temporel h_t, que nous présentons sur les schémas de la figure 5.21.

5.6.3 Résultats

Un entraînement plus court

L'entraînement complet de la méthode STaRFlow prend environ deux semaines [5]. Afin de pouvoir obtenir des résultats plus rapidement, nous proposons d'entraîner ces méthodes selon le même protocole (FlyingChairs → FlyingThings3D) que précédemment mais en divisant par deux le nombre d'itérations réalisées sur chaque jeu de données.

Résultats quantitatifs sur Sintel

Le tableau 5.9 donne les erreurs moyennes sur Sintel pour ces méthodes entraînées avec l'entraînement court, nous avons également réentraîné STaRFlow avec cet entraî-

5. Ceci est détaillé en annexe dans la section A.5.

TABLEAU 5.9 – Erreur *endpoint* moyenne obtenue avec différentes méthodes après entraînement court sur FlyingChairs → FlyingThings3D. Les meilleurs résultats sont en caractères gras.

Sans discontinuités temporelles pendant l'entraînement

Méthode	Sintel Clean			Sintel Final		
	all	noc	occ	all	noc	occ
STaRFlow	1.79	1.01	10.22	**3.15**	2.04	**15.14**
GatedSTaRFlow	1.89	1.07	10.75	3.19	2.06	15.36
GRU-early	**1.77**	**0.97**	10.48	3.16	**2.02**	15.49
GRU-late	1.79	1.01	**10.19**	3.21	2.06	15.58

Avec discontinuités temporelles pendant l'entraînement

Méthode	Sintel Clean			Sintel Final		
	all	noc	occ	all	noc	occ
STaRFlow	**1.63**	**0.88**	**9.76**	**3.14**	2.04	**15.08**
GatedSTaRFlow	1.74	0.95	10.24	3.20	2.04	15.75
GRU-early	1.72	0.92	10.36	**3.14**	**1.98**	15.71
GRU-late	1.68	0.89	10.14	3.21	2.04	15.88

nement plus court pour pouvoir réaliser une comparaison juste, à nombre d'itérations identique. On remarque que les autres architectures proposées n'apportent globalement pas d'amélioration notable sur l'erreur moyenne. L'ajout de discontinuités temporelles pendant l'entraînement conduit à une amélioration très légère du résultat.

Résultat qualitatif sur une discontinuité temporelle

La figure 5.22 compare ces différentes méthodes sur l'exemple de la figure 5.19 qui nous a permis de mettre en évidence le problème des discontinuités temporelles. Sans discontinuités temporelles pendant l'entraînement, aucune des architectures ne parvient à estimer correctement le mouvement — qui change brutalement — de la tête du personnage de gauche. En ajoutant des discontinuités temporelles aux données d'entraînement, STaRFlow et GRU-late parviennent à proposer une estimation correcte. Ceci met en évidence l'importance des phénomènes présentés dans les données d'entraînement, et montre que l'architecture STaRFlow est suffisante pour concevoir une méthode permettant d'implémenter un mécanisme d'oubli permettant de gérer les discontinuités temporelles, du moment que de telles discontinuités sont présentées pendant l'entraînement.

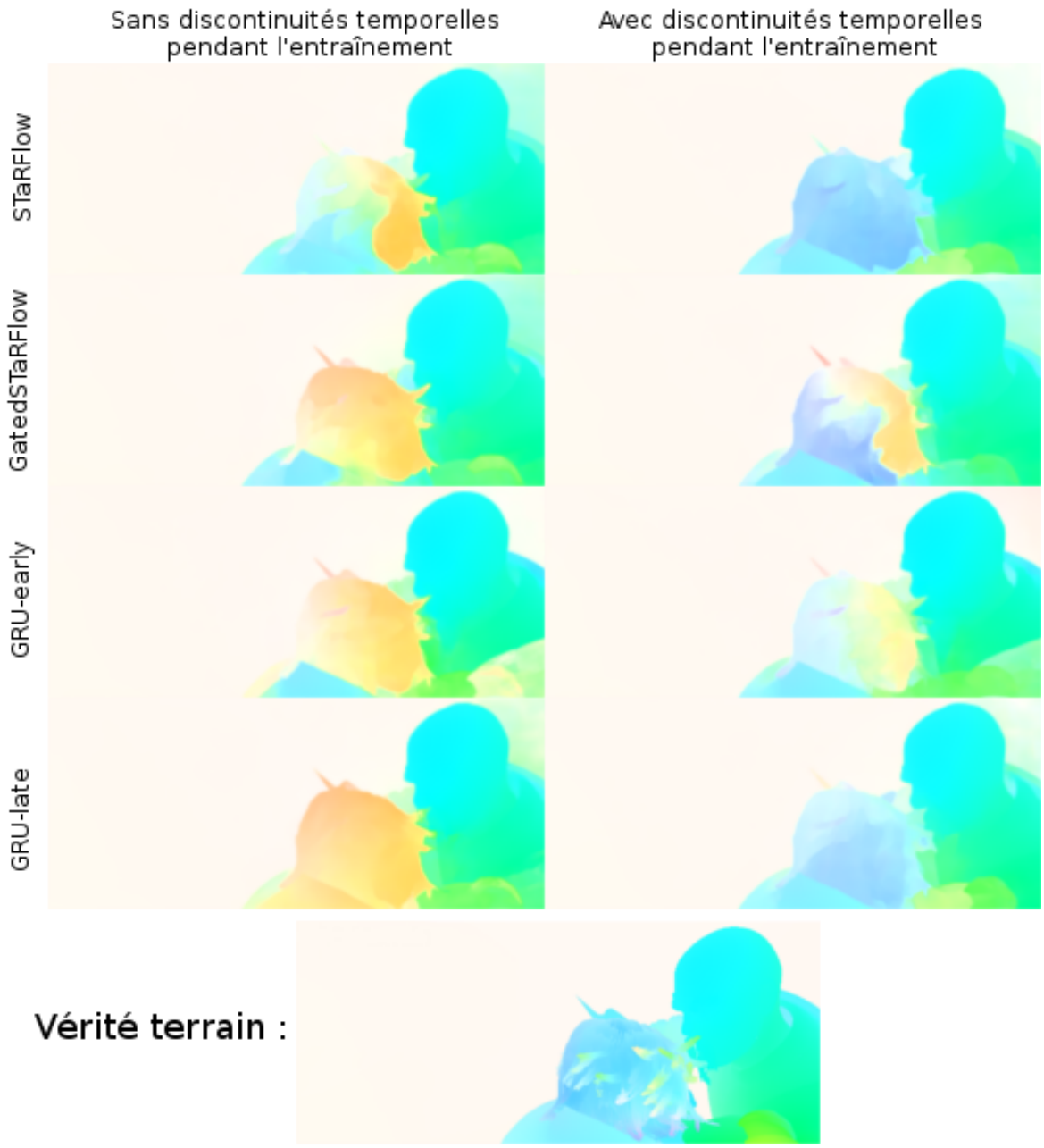

FIGURE 5.22 – Comparaison des méthodes avec entraînement court, pour la 46e paire d'images de la séquence "Ambush_5" de Sintel Clean.

5.7 Publication et disponibilité du code

Les travaux sur l'architecture STaRFlow ont été acceptés à la conférence ICPR 2020 (GODET et collab. [2020]).

L'implémentation en Python de STaRFlow est en ligne à l'adresse
https://github.com/pgodet/star_flow.

5.8 Conclusion du chapitre

Nous avons proposé un algorithme récurrent permettant de réaliser une estimation multiframe du flot optique en transmettant un état de mémoire constitué de cartes d'activations apprises d'un instant au suivant. Nous avons obtenu avec cette connexion temporelle de meilleurs résultats qu'avec celle proposée par NEORAL et collab. [2018] (section 5.3.2), et permettant, de plus, d'exploiter une continuité temporelle à plus long terme (section 5.3.2).

Nous avons pu montrer plusieurs intérêts d'une estimation tenant compte du passé, par comparaison aux architectures biframe PWC-Net et IRR-PWC, dans les sections 5.3.2 et 5.5.3. Exploiter l'information temporelle permet d'améliorer la qualité de l'estimation du flot optique dans le cas d'images dégradées, de petits objets ou d'occultations. Nous avons également observé de meilleures capacités de généralisation sur KITTI 2015, par rapport à la méthode IRR-PWC proposée par HUR et ROTH [2019] (section 5.5.3).

Nos expériences en biframe, dans la section 5.2.2, montrent que l'estimation du flot optique et la détection des occultations, sont des tâches suffisamment proches pour être réalisées avec un même réseau partageant la quasi-totalité des poids pour les deux tâches. Dans notre méthode, la séparation entre ces deux tâches ne se fait qu'à la toute dernière couche de convolution de l'estimateur. Par ailleurs nos expériences montrent, dans le cas biframe et dans le cas multiframe, que l'ajout de la détection des occultations améliore l'estimation de flot optique (section 5.3.2). Ce résultat confirme les conclusions de NEORAL et collab. [2018] et HUR et ROTH [2019] à ce sujet.

En partageant les mêmes poids pour l'estimation aux différents niveaux d'échelle, comme proposé par HU et collab. [2018] et HUR et ROTH [2019], et pour les deux tâches d'estimation du flot et de détection des occultations, notre méthode permet d'utiliser moins de paramètres que les autres méthodes par apprentissage profond, pour obtenir des résultats à l'état de l'art sur les *benchmarks* MPI Sintel et KITTI 2015. La figure 5.23 place STaRFlow parmi les autres méthodes de l'état de l'art en termes de précision sur le *benchmark* MPI Sintel Final et de taille du modèle en nombre de paramètres.

L'ajout du module de *refinement* proposé par HUR et ROTH [2019] permet une amélioration significative des résultats. Nous avons quantifié cet apport dans le cas de STaRFlow dans la section 5.3.3. Cet ajout conduit à une amélioration des résultats de 11 % sur Sintel Final, de 22 % sur Sintel Clean et de 27% sur KITTI, pour une augmentation de la taille du modèle inférieure à 10 %. Cependant, à force d'ajout de nouveaux modules, le modèle se complique peu à peu, avec un premier estimateur de flot incluant des connexions denses, suivi d'un *context network* réalisant des convolutions dilatées (qui était appelé une seule fois, à la fin, dans PWC-Net et qui est appelé à chaque niveau d'échelle depuis IRR-PWC), auquel on ajoute maintenant un module de *refinement* réalisant des convolutions par des filtres spécifiques pour chaque voisinage spatial, estimés par un autre petit réseau. Cette construction par accumulation a permis l'amélioration successive des résultats depuis l'architecture PWC-Net, mais on peut maintenant se poser la question de trouver une version plus compacte de l'architecture. En effet, si STaRFlow permet d'obtenir de très

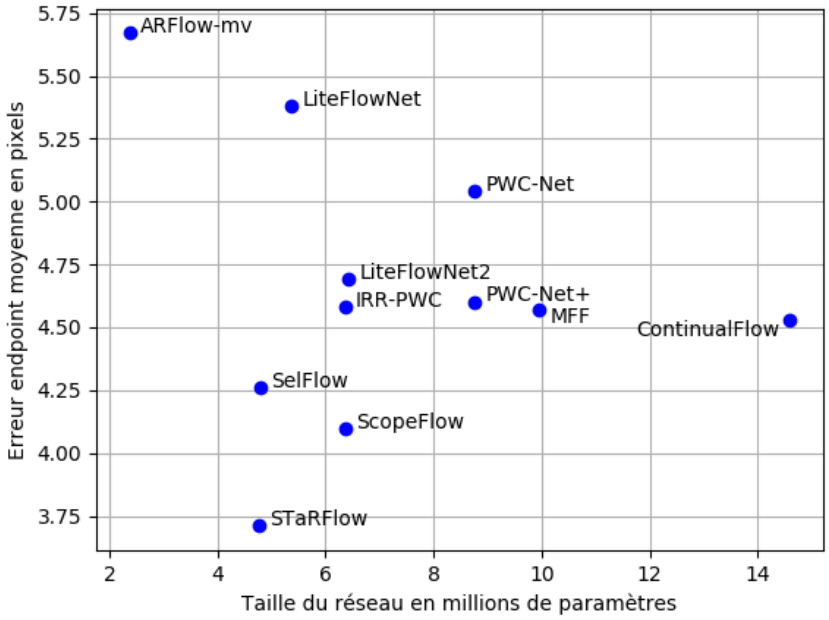

FIGURE 5.23 – Comparaison de STaRFlow aux méthodes de l'état de l'art (publiées avant juillet 2020) pour l'estimation de flot optique par apprentissage profond, en termes de nombre de paramètres appris et d'erreur *endpoint* moyenne sur le *benchmark* MPI Sintel Final.

bons résultats, son occupation mémoire et son temps de calcul (bien que nettement plus rapide que les meilleures méthodes variationnelles) demeurent trop importants pour une utilisation dans un contexte embarqué. Supprimer ou alléger certains modules allégerait le nombre de paramètres et réduirait également les temps de calcul.

Nous avons, enfin, mis en évidence le problème des discontinuités temporelles. Un changement brusque dans une trajectoires peut induire STaRFlow en erreur. Pour disposer d'une méthode capable de gérer la détection des cas où il est préférable d'oublier l'information du passé, il faut introduire des cas de rupture de la continuité temporelle dans les données d'entraînement. Pour cela, on peut commencer par créer des inversions temporelle dans les séquences, pour créer des aller-retours, mais peut-être faut-il aller plus loin et générer des exemples où certains objets présentent des changement brusques de trajectoire tandis que d'autres bougent plus calmement, comme il est probable de rencontrer en pratique, comme dans l'exemple de Sintel présenté sur la figure 5.19.

5.9 Références

BALLAS, N., L. YAO, C. PAL et A. COURVILLE. 2015, «Delving deeper into convolutional networks for learning video representations», *arXiv preprint arXiv :1511.06432*. 127

CHERRIER, N., T. CASTAINGS et A. BOULCH. 2017, «Deep sequence-to-sequence neural networks for ionospheric activity map prediction», dans *International Conference on Neural Information Processing*, Springer, p. 545–555. 127

DOSOVITSKIY, A., P. FISCHER, E. ILG, P. HAUSSER, C. HAZIRBAS, V. GOLKOV, P. VAN DER SMAGT, D. CREMERS et T. BROX. 2015, «FlowNet : Learning optical flow with convolutional networks», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 2758–2766. 102

GODET, P., A. BOULCH, A. PLYER et G. LE BESNERAIS. 2020, «Starflow : A spatiotemporal recurrent cell for lightweight multi-frame optical flow estimation», dans *25th International Conference on Pattern Recognition, ICPR 2020, Milan, Italy, 10-15 janvier 2021*, IEEE Computer Society. 132

HU, P., G. WANG et Y.-P. TAN. 2018, «Recurrent spatial pyramid CNN for optical flow estimation», *IEEE Transactions on Multimedia*, vol. 20, nº 10, p. 2814–2823. 132

HUI, T.-W., X. TANG et C. C. LOY. 2018, «LiteFlowNet : A lightweight convolutional neural network for optical flow estimation», dans *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 121

HUR, J. et S. ROTH. 2019, «Iterative residual refinement for joint optical flow and occlusion estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5754–5763. x, 102, 103, 104, 105, 107, 108, 110, 114, 115, 117, 118, 119, 120, 121, 132

ILG, E., N. MAYER, T. SAIKIA, M. KEUPER, A. DOSOVITSKIY et T. BROX. 2017, «FlowNet 2.0 : Evolution of optical flow estimation with deep networks», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 6. 119

ILG, E., T. SAIKIA, M. KEUPER et T. BROX. 2018, «Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation», dans *European Conference on Computer Vision*, Springer, p. 626–643. 103

LIU, L., J. ZHANG, R. HE, Y. LIU, Y. WANG, Y. TAI, D. LUO, C. WANG, J. LI et F. HUANG. 2020, «Learning by analogy : Reliable supervision from transformations for unsupervised optical flow estimation», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 6489–6498. 121

LIU, P., M. R. LYU, I. KING et J. XU. 2019, «SelFlow : Self-supervised learning of optical flow», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 121

MAYER, N., E. ILG, P. HAUSSER, P. FISCHER, D. CREMERS, A. DOSOVITSKIY et T. BROX. 2016, «A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4040–4048. 102

NEORAL, M., J. ŠOCHMAN et J. MATAS. 2018, «Continual occlusion and optical flow estimation», dans *Asian Conference on Computer Vision*, Springer, p. 159–174. 102, 103, 105, 107, 109, 110, 115, 119, 132

REN, Z., O. GALLO, D. SUN, M.-H. YANG, E. SUDDERTH et J. KAUTZ. 2019, «A fusion approach for multi-frame optical flow estimation», dans *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, p. 2077–2086. 121

REVAUD, J., P. WEINZAEPFEL, Z. HARCHAOUI et C. SCHMID. 2015, «Epicflow : Edge-preserving interpolation of correspondences for optical flow», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 1164–1172. 123

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018a, «Models matter, so does training : An empirical study of CNNs for optical flow estimation», *arXiv preprint arXiv :1809.05571*. 102, 119, 120

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018b, «PWC-Net : CNNs for optical flow using pyramid, warping, and cost volume», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 8934–8943. 108, 110, 118

XINGJIAN, S., Z. CHEN, H. WANG, D.-Y. YEUNG, W.-K. WONG et W.-C. WOO. 2015, «Convolutional lstm network : A machine learning approach for precipitation nowcasting», dans *Advances in neural information processing systems*, p. 802–810. 127

ZHAO, S., Y. SHENG, Y. DONG, E. I. CHANG, Y. XU et collab.. 2020, «MaskFlownet : Asymmetric feature matching with learnable occlusion mask», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 6278–6287. 107

Chapitre 6

Généralisation et comparaison avec les méthodes classiques

Sommaire

6.1 Introduction

La méthode d'estimation multiframe de flot optique par apprentissage profond, STaR-Flow, que nous avons proposée et évaluée dans le chapitre 5, donne des résultats à l'état de l'art sur les *benchmarks* MPI Sintel et KITTI 2015. Ces résultats ont été obtenus après une spécialisation de la méthode sur le contexte ciblé, par *finetuning* sur l'ensemble d'entraînement de MPI Sintel ou de KITTI avant évaluation sur l'ensemble de test. Ici, nous proposons de tester les méthodes STaRFlow et IRR-PWC (HUR et ROTH [2019]) à l'issue de l'entraînement FlyingChairs → FlyingThings3D, sur des données issues de contextes variés afin d'évaluer leur capacité de généralisation. Si ces méthodes dépassent largement les méthodes classiques sur les *benchmarks* Sintel et KITTI (après *finetuning*), qu'en est-il dans le cas d'une utilisation réelle, sans *finetuning* ? Pour répondre à cette question, nous comparons les résultats de ces méthodes par apprentissage profond, IRR-PWC et STaR-Flow, sans *finetuning*, à ceux de la méthode locale FOLKI (LE BESNERAIS et CHAMPAGNAT [2005]), et à ceux de la méthode globale DeepFlow (WEINZAEPFEL et collab. [2013]).

6.1.1 Choix des séquences pour l'évaluation

Pour évaluer ces méthodes nous avons choisi plusieurs séquences, présentant des contextes variés et différents de celui des données d'entraînement (chaises animées de mouvements 2D puis modèles 3D d'objets animés de mouvements 3D dans un environnement simulé). Certaines séquences sont issues de données réelles, d'autres sont simulées. Nous les avons regroupées selon trois catégories.

— La première catégorie regroupe des séquences mettant en scène des objets mobiles, afin d'évaluer les performances des méthodes pour des applications comme la reconnaissance d'action, ou la navigation basée vision (à travers des tâches comme le suivi d'objets mobiles, la prédiction de trajectoires ou la détection de piétons).

— La deuxième catégorie contient des mouvements globaux de la scène, dus au mouvement propre de la caméra : elle correspond à des problématiques d'odométrie visuelle (dont le but est la mesure de la trajectoire de la caméra) ou de reconstruction 3D par stéréo-mouvement.

— La dernière catégorie concerne le contexte de la mesure par imagerie de champs de vitesses pour la mécanique, soit dans le cas de la déformation d'un matériau sous contrainte mécanique, soit en vélocimétrie par images de particules en mécanique des fluides.

Chaque contexte applicatif présente des exigences différentes quant à la qualité de l'estimation de mouvement. Le contexte de la navigation et la reconnaissance d'action gagneront à disposer d'un flot dense et dans lequel les objets sont bien segmentés par rapport au fond. Dans le cas de la détection de petits objets mobiles (par exemple un piéton ou un véhicule éloigné de la caméra) la résolution spatiale est cruciale. Enfin, la mesure par imagerie nécessitera une estimation dense et qui soit particulièrement précise (l'erreur sur le déplacement mesuré devant être très nettement inférieure à la taille du pixel dans ce contexte).

Dans la suite, nous présentons les flots estimés par les différentes méthodes en utilisant la représentation colorée présentée dans le chapitre 2, en section 2.1.1. Nous rappelons la correspondance teinte-direction sur la figure 6.1.

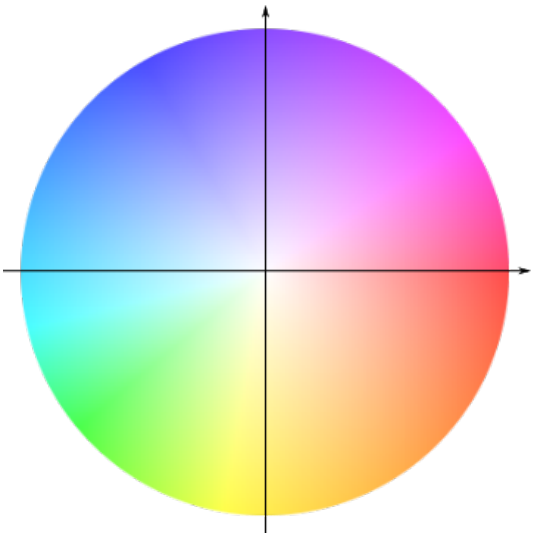

FIGURE 6.1 – Correspondance teinte-direction de la représentation colorée.

6.1.2 Implémentation et paramétrage des méthodes

FOLKI

Nous utilisons la forme inverse de FOLKI, décrite dans le chapitre 2, section 2.2.1. Cette méthode ne tient compte que de deux images consécutives. Il y a trois paramètres à régler :

— le rayon de la fenêtre, qui est uniforme et carrée de côté $2 \times \text{rayon} + 1$;

— le nombre de niveaux de pyramide pour l'estimation multi-échelle (les dimensions sont divisées par 2 entre deux niveaux, et le nombre de niveaux indiqué n'inclut pas la pleine résolution, pour 3 niveaux on commence donc l'estimation sur des images sous-échantillonnées d'un facteur 8) ;

— le nombre d'itérations (pour l'algorithme itératif de Gauss-Newton) par niveau de pyramide.

Le rayon et le nombre de niveaux utilisés seront indiqués pour chaque exemple. Le nombre d'itérations est systématiquement 5.

DeepFlow

Nous utilisons l'implémentation en langage C des auteurs (DeepFlow2), disponible à l'adresse https://thoth.inrialpes.fr/src/deepflow/. DeepFlow utilise une étape de *matching* de descripteurs, avec l'algorithme DeepMatching, avant de minimiser une énergie variationnelle (selon une méthode proche de celle proposée par BROX et collab. [2004], mais ajoutant un terme prenant en compte le résultat du *matching*) pour obtenir le flot optique. Cette étape de *matching* permet de mesurer les très grands mouvements. Nous utilisons l'implémentation en langage C de DeepMatching, disponible à l'adresse https://thoth.inrialpes.fr/src/deepmatching/. DeepFlow ne tient compte que de deux images consécutives.

Dans la suite, nous utiliserons parfois les paramètres par défaut de DeepFlow, et parfois le jeu de paramètres utilisés par les auteurs lors de l'évaluation sur MPI Sintel. Les paramètres sont les suivants :

— α : pondération du terme de régularisation ;

TABLEAU 6.1 – Choix des paramètres de pondération et de lissage pour l'algorithme DeepFlow. Jeu de paramètres par défaut et jeu de paramètres utilisé pour l'évaluation sur Sintel.

	α	β	γ	δ	σ	b
Par défaut	1	20	0.7	0.1	0.6	0.5
Sintel	1	32	0.56	0	0.45	0.45

— β : pondération du terme associé au *matching* ;

— γ : pondération du terme de conservation des gradients dans l'attache aux données ;

— δ : pondération du terme de conservation de la couleur dans l'attache aux données ;

— σ : les images d'entrées sont lissées en utilisant un noyau gaussien d'écart-type σ ;

— $b \in [0;1]$: à chaque niveau de pyramide k, la valeur de la pondération associée au *matching* est en fait différente et vaut $\beta(k) = \beta \times (k/k_{max})^b$ où k_{max} désigne le niveau le plus grossier, ainsi on donne moins d'importance au *matching* aux échelles fines ;

— $\eta = 0.95 \in [0;1]$: facteur de sous-échantillonnage entre deux niveaux de pyramide consécutifs ;

— `minsize` = 25 pixels : taille du niveau de pyramide le plus grossier ;

— `inner` = 5 : nombre d'itérations pour l'algorithme du point fixe, par niveau de pyramide ;

— `iter` = 25 : nombre d'itérations du minimiseur SOR ("Successive Over Relaxation", ou "Méthode de surrelaxation successive") ;

— $\omega = 1.6$: paramètre de relaxation pour l'algorithme SOR.

Le tableau 6.1 donne les valeurs des paramètres de pondération et de lissage par défaut et ceux utilisés par les auteurs pour évaluer l'algorithme sur Sintel. Les paramètres du minimiseur et ceux concernant la pyramide multi-échelle sont communs aux deux cas. Parfois nous n'utilisons pas l'étape de *matching* (notamment s'il n'y a pas de grands mouvements), dans ce cas la pondération β est mise à 0.

IRR-PWC

Nous utilisons l'architecture et les poids issus de l'entraînement sur FlyingChairsOcc → FlyingThings3Dsubset. Tout est rendu disponible par les auteurs à l'adresse https://github.com/visinf/irr. Cette méthode ne tient compte que de deux images consécutives.

STaRFlow

STaRFlow est entraîné comme indiqué dans le chapitre 5, en se limitant à FlyingChairsOcc → FlyingThings3Dsubset. Sur FlyingThings3D, l'entraînement multiframe est réalisé en considérant des séquences de $N = 4$ images. Dans la suite, lorsque rien n'est précisé, la méthode est évaluée en considérant $N' = 4$ images consécutives. Quel que soit le nombre d'images N' utilisé pour l'évaluation, le flot d'intérêt est celui entre l'avant-dernière et la dernière image, augmenter l'horizon temporel pour cette méthode signifie donc ajouter des images du passé.

6.2 Estimation du mouvement d'objets mobiles

Nous analysons dans cette section les résultats obtenus sur quatre séquences réelles issues de contextes différents, en nous intéressant particulièrement à l'estimation du mouvement des objets mobiles.

6.2.1 Séquence de jonglage

La première séquence que nous considérons présente une scène de jonglage. Les images et résultats donnés par les différentes méthodes sont présentés sur la figure 6.2. Il s'agit d'une séquence réelle, filmée avec un téléphone portable. Les balles et les différentes parties du corps sont des objets mobiles. Les balles sont un cas particulier de petits objets présentant des déplacement de grande amplitude, connu pour poser problème aux approches classiques utilisant une estimation multi-échelle pyramidale. La caméra bouge légèrement vers la droite, induisant un mouvement global du fond, de faible amplitude, vers la gauche.

Remarquons premièrement que les méthodes par apprentissage profond, IRR-PWC et STaRFlow, généralisent très bien à cette séquence réelle, alors qu'elles n'ont été entraînées, rappelons-le, que sur des données synthétiques. Soulignons en particulier la cohérence des estimations proposées par les quatre méthodes, tant en termes de direction qu'en termes d'amplitude des mouvements estimés.

L'estimation de FOLKI présente un aspect bruité sur le fond, et une estimation peu précise voire erronée proche des discontinuités du champ de mouvement. On obtient avec DeepFlow un flot nettement plus lisse (par morceau), mais qui semble peu précis au niveau des discontinuités spatiales. Un halo blanc apparaît notamment à droite du corps, là où les points du fond sont occultés dans l'image cible. Les méthodes IRR-PWC et STaRFlow donnent un résultat beaucoup plus propre sur les discontinuités spatiales, permettant une meilleure segmentation des objets. Ces discontinuités sont encore plus marquées dans le résultat de STaRFlow, menant à une estimation d'aspect très propre.

6.2.2 Séquence du coureur

Nous reprenons ici la séquence du coureur présentée dans le chapitre 4. Les résultats obtenus sont présentés sur la figure 6.3. Il s'agit également d'une séquence réelle et, là encore, les méthodes utilisant l'apprentissage profond généralisent bien puisqu'elles proposent des directions et amplitudes des mouvements du fond et des parties du corps cohérentes avec celles estimées par les méthodes classiques. Les méthodes IRR-PWC et STaRFlow sont, sur cette séquence également, meilleures en termes de segmentation du corps par rapport au fond. STaRFlow donne une estimation légèrement meilleure que celle de IRR-PWC sur le haut du corps, mais les marqueurs blancs sur la jambes posent problème à STaRFlow, pour une raison qui nous échappe. Remarquons enfin que les résultats obtenus avec STaRFlow sont bien plus propres que ceux que nous avions obtenus dans le chapitre 4 avec PWCNetStack.

6.2.3 Séquence du piéton

Nous considérons maintenant, sur la figure 6.4, une séquence réelle issue du jeu de données nuScenes (CAESAR et collab. [2019]), représentant une scène de navigation routière, au niveau d'une intersection où un piéton traverse, deux voitures se croisent derrière

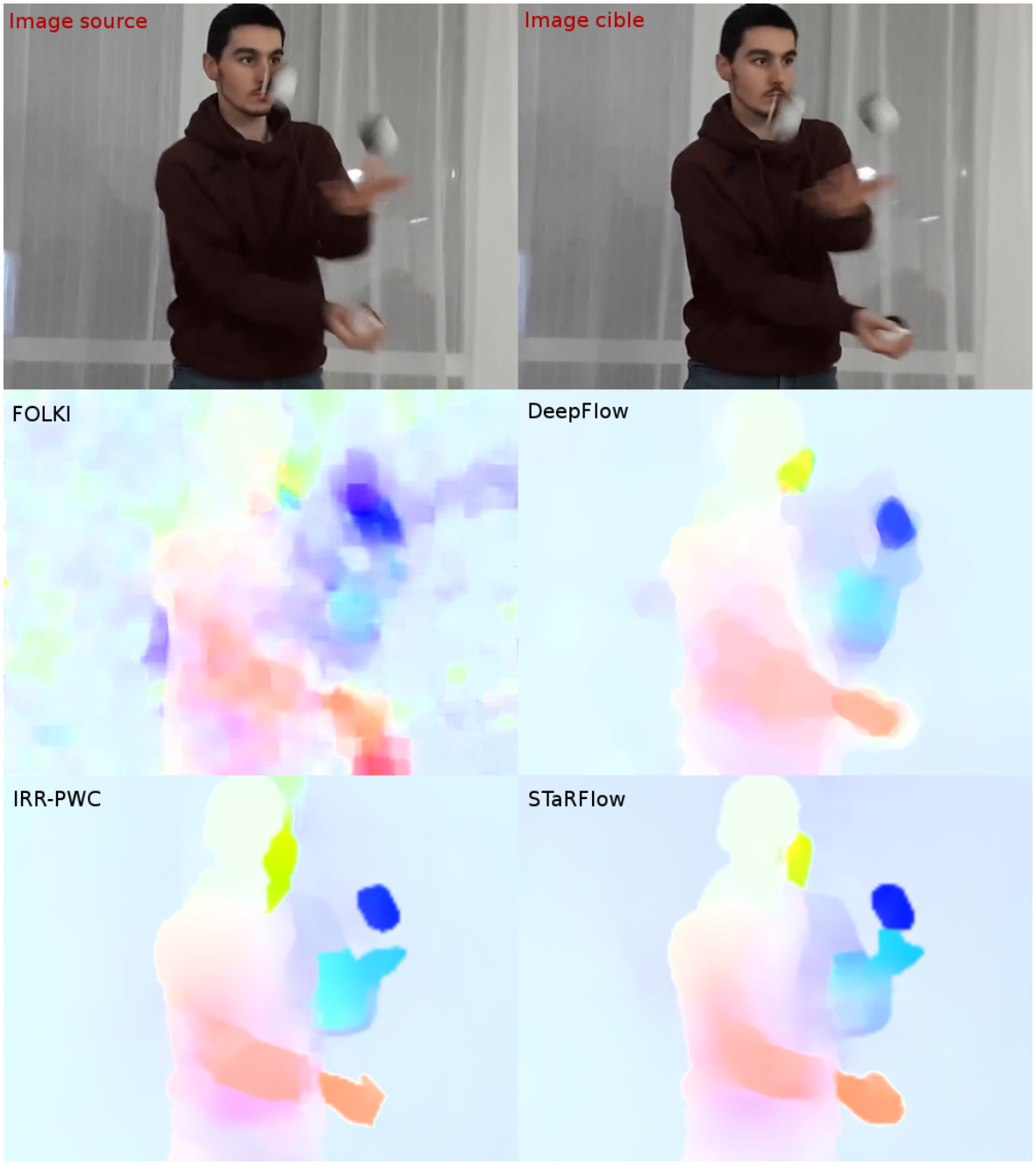

FIGURE 6.2 – Estimation du flot optique sur la séquence de jonglage par les méthodes FOLKI (3 niveaux, rayon de la fenêtre = 10 pixels), DeepFlow (avec *matching* et jeu de paramètres "Sintel"), IRR-PWC et STaRFlow ($N' = 4$).

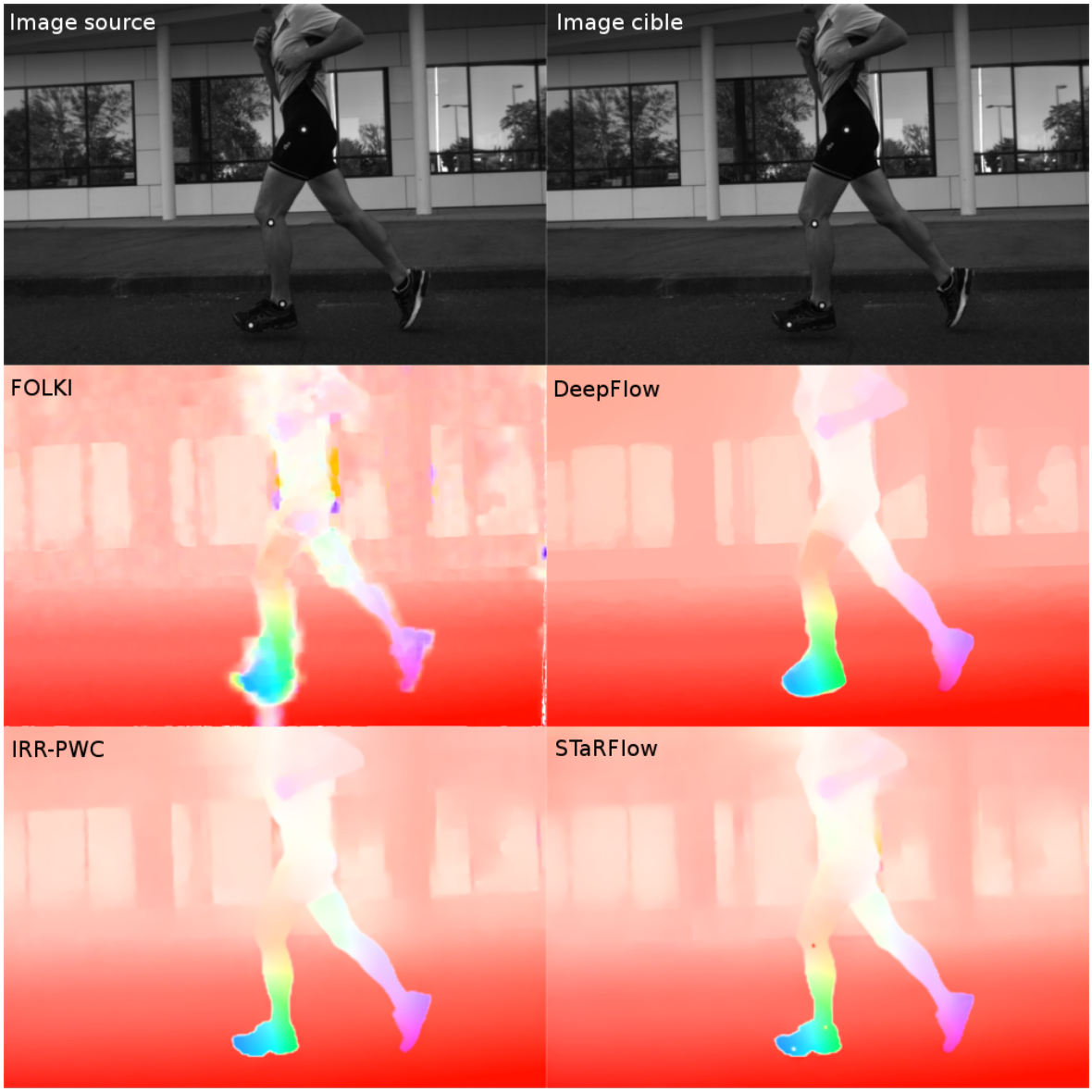

FIGURE 6.3 – Estimation du flot optique sur la séquence du coureur par les méthodes FOLKI (3 niveaux, rayon de la fenêtre = 10 pixels), DeepFlow (avec *matching* et jeu de paramètres "Sintel"), IRR-PWC et STaRFlow ($N' = 4$).

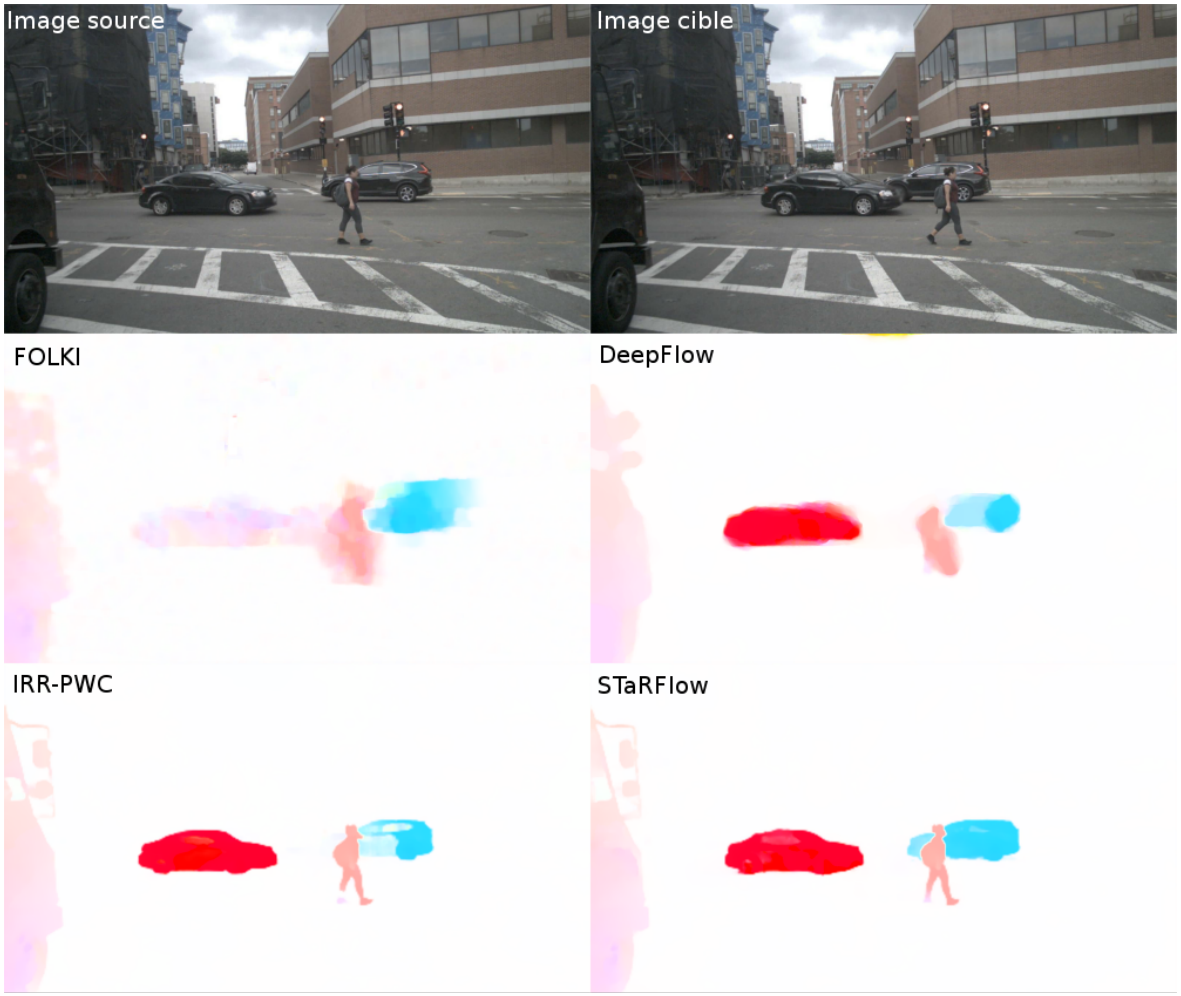

FIGURE 6.4 – Estimation du flot optique sur la séquence du piéton par les méthodes FOLKI (5 niveaux, rayon de la fenêtre = 10 pixels), DeepFlow (avec *matching* et jeu de paramètres "Sintel"), IRR-PWC et STaRFlow ($N' = 4$).

le piéton, et un camion s'avance à gauche du champ de vue en direction de l'intersection. La caméra filmant la scène est immobile. Cette scène présente des grands mouvements d'objets mobiles, en particulier le déplacement de la voiture de gauche, qui est trop ample pour l'algorithme FOLKI. Une autre difficulté de cet exemple est l'occultation partielle de la voiture de droite par le piéton puis par l'autre voiture. Enfin, cette scène présente des structures spatiales fines, comme les jambes du piéton ou le rétroviseur du camion à gauche du champ.

Les méthodes IRR-PWC et STaRFlow généralisent bien à ce contexte, et donnent de meilleurs résultats en termes de finesse spatiale et de segmentation des objets que les méthodes FOLKI et DeepFlow, ce que l'on peut voir notamment sur le rétroviseur du camion de gauche et sur les jambes du piéton. STaRFlow est la seule méthode à proposer une estimation correcte au niveau de l'avant de la voiture de droite. Il s'agit là d'un cas difficile, dans la mesure où l'avant de la voiture est partiellement occulté dans les deux images d'intérêt, des zones différentes étant cachées dans chacune des deux images, ce qui implique à la fois des cas d'occultations et de désoccultations.

6.2.4 Séquence de la foule dans la gare

Afin d'évaluer les méthodes dans un contexte de surveillance vidéo, dans le cas d'une foule en mouvement, nous utilisons une séquence filmée dans une gare, trouvée sur YouTube. Les images et résultats sont présentés sur la figure 6.5. Il s'agit une fois de plus d'une séquence réelle montrant les capacités de généralisation de STaRFlow et IRR-PWC.

Les estimations obtenues avec IRR-PWC et STaRFlow permettent de mieux détourer les objets que celles obtenues avec FOLKI ou DeepFlow. Il n'est cependant pas certain que STaRFlow donne un meilleur résultat que celui obtenu avec IRR-PWC sur cet exemple. Par ailleurs, ces deux méthodes donnent un mouvement non nul à certains endroits du sol, là où DeepFlow ne détecte aucun mouvement. La séquence présente des reflets et des effets de blocs dus à la compression, qui trompent probablement les méthodes par apprentissage.

Sur cette séquence, avec de multiples petits objets mobiles, il semble y avoir encore une marge d'amélioration possible par rapport aux résultats obtenus avec IRR-PWC et STaRFlow. Le résultat semble peu résolu spatialement, ce qui suggère d'étudier la question d'une estimation à pleine résolution avec ces méthodes. Rappelons que ces dernières estiment le flot au quart de la résolution des images d'entrée puis réalisent une interpolation bilinéaire pour obtenir un flot à pleine résolution.

FIGURE 6.5 – Estimation du flot optique sur la séquence de la foule dans la gare par les méthodes FOLKI (3 niveaux, rayon de la fenêtre = 8 pixels), DeepFlow (avec *matching* et jeu de paramètres "Sintel"), IRR-PWC et STaRFlow ($N' = 4$).

6.3 Estimation du mouvement global

Intéressons-nous maintenant, dans le contexte de la navigation basée vision, à l'estimation du mouvement global dans le but, par exemple, de mesurer la trajectoire de la caméra ou de cartographier l'environnement 3D. Pour cela nous considérons deux séquences extraites du jeu de données de synthèse TartanAir (WANG et collab. [2020]), dont le but est d'évaluer et comparer des méthodes de SLAM visuel (SLAM signifie "Simultaneous Localization and Mapping", ou "Cartographie et localisation simultanées"), et une séquence issue du film *Star Wars : Episode IV – A New Hope* présentant un mouvement global convergent. Les séquences de TartanAir sont annotées de la vérité terrain du flot optique, permettant de réaliser une évaluation quantitative des estimations obtenues. La séquence de *Star Wars* présente un mouvement global convergent dont l'allure est facilement reconnaissable sur la représentation colorée, permettant une évaluation qualitative simple.

6.3.1 Séquence nocturne de l'usine abandonnée (TartanAir)

La première séquence extraite du jeu de données TartanAir est issue de la séquence appelée "AbandonnedFactory_night", les images présentent une illumination faible, ce qui rend cette séquence difficile. Sur cette séquence, pour laquelle les résultats sont donnés sur la figure 6.6, STaRFlow et IRR-PWC donnent quantitativement des résultats nettement meilleurs que ceux de FOLKI et DeepFlow. L'analyse visuelle des flots optiques estimés montre que STaRFlow donne un meilleur résultat dans la zone d'ombre en haut à droite, là où toutes les autres méthodes donnent un résultat erroné. Ce résultat indique un très bon potentiel de STaRFlow pour son utilisation dans un contexte de navigation basée vision dans un cas de faible illumination.

6.3.2 Séquence sous la pluie (TartanAir)

La seconde séquence extraite du jeu de données TartanAir est issue de la séquence appelée "SoulCity". Il s'agit d'une séquence où il pleut. En plus des gouttes de pluie qui tombent, cela donne lieu à des reflets et réflexions spéculaires sur certaines surfaces mouillées, comme sur le sol du pont, derrière la barrière qui est au premier plan. Les images et résultats sont présentés sur la figure 6.7. Qualitativement, le résultat de STaRFlow est très impressionnant sur les barreaux de la barrière, qui sont des détails spatiaux très fins et qui ressortent de manière très visible sur l'estimation du mouvement, uniquement avec STaRFlow. Quantitativement, DeepFlow, IRR-PWC et STaRFlow présentent des erreurs moyennes proches les unes des autres, et nettement inférieures à celle de FOLKI. La méthode STaRFlow est quantitativement légèrement moins précise que la méthode IRR-PWC, notamment sur le sol mouillé du pont derrière la barrière, et à cause des gouttes de pluie qui sont ignorées par IRR-PWC mais visibles sur l'estimation de STaRFlow.

La vérité terrain proposée donne la projection dans le plan de la caméra du mouvement 3D des objets, ignorant donc les gouttes de pluie, et les réflexions spéculaires sur les surfaces mouillées. Au contraire, une méthode d'estimation de flot optique estime le mouvement *apparent*. Ceci pose la question de ce que l'on cherche réellement à estimer. Il est possible que l'estimation de STaRFlow soit plus juste en termes de mouvement apparent, mais jugée plus fausse par rapport à la vérité terrain calculée à partir du mouvement 3D des objets. Les données d'entraînement vues par STaRFlow et IRR-PWC ne contiennent pas de reflets ou de réflexions spéculaires, dans ces données le mouvement

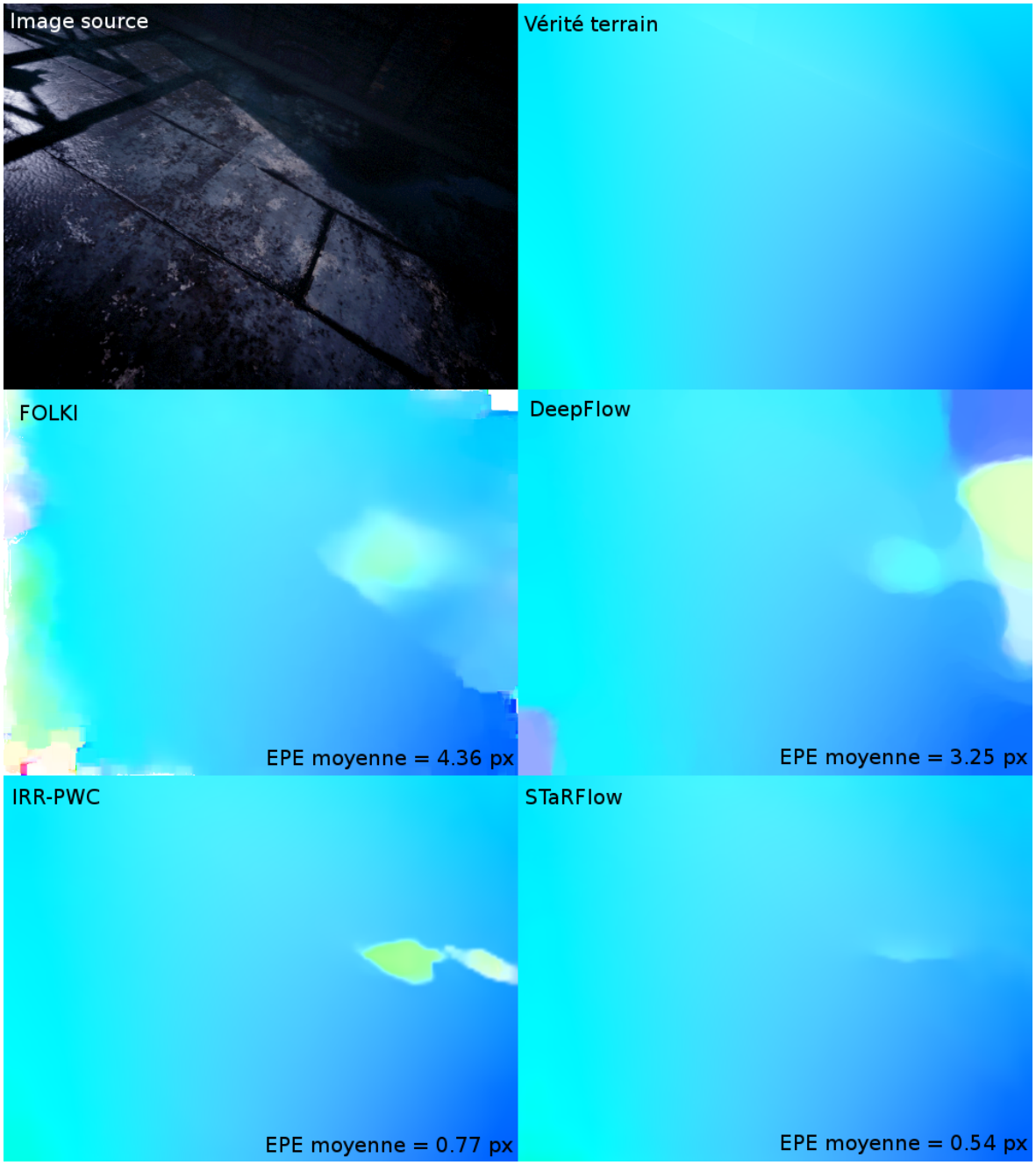

FIGURE 6.6 – Estimation du flot optique sur la séquence nocturne par les méthodes FOLKI (3 niveaux, rayon de la fenêtre = 10 pixels), DeepFlow (avec *matching* et jeu de paramètres "Sintel"), IRR-PWC et STaRFlow ($N' = 4$).

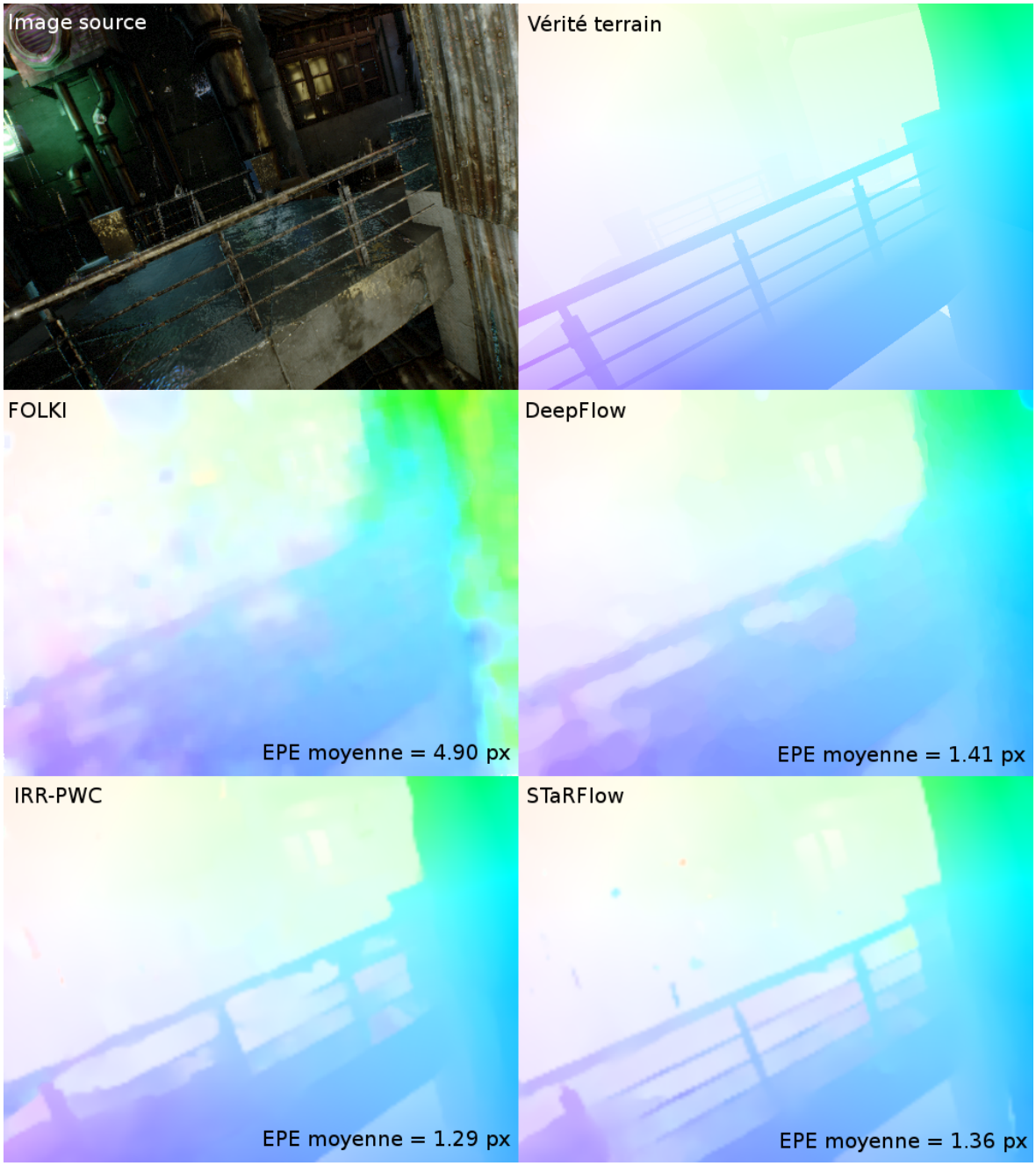

FIGURE 6.7 – Estimation du flot optique sur la séquence sous la pluie par les méthodes FOLKI (4 niveaux, rayon de la fenêtre = 7 pixels), DeepFlow (avec *matching* et jeu de paramètres par défaut), IRR-PWC et STaRFlow ($N' = 4$).

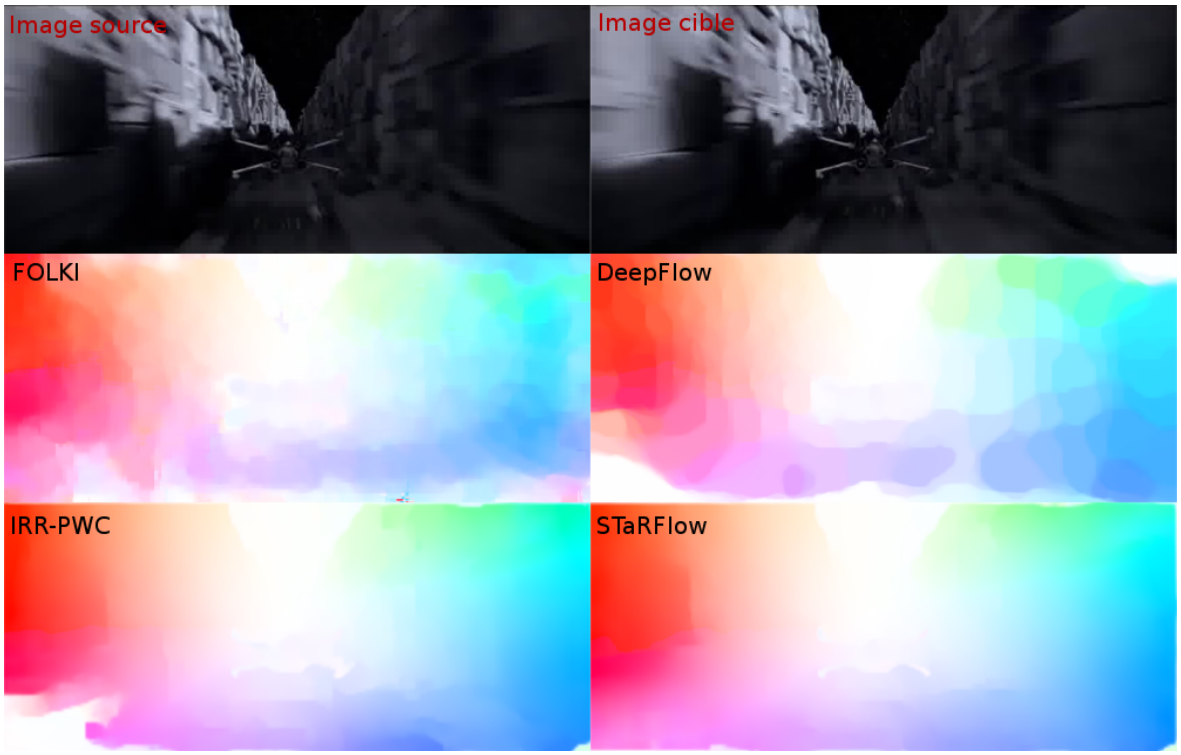

FIGURE 6.8 – Estimation du flot optique sur la séquence issue de Star Wars par les méthodes FOLKI (3 niveaux, rayon de la fenêtre = 10 pixels), DeepFlow (avec *matching* et jeu de paramètres "Sintel"), IRR-PWC et STaRFlow ($N' = 4$).

apparent correspond bien à la projection sur la caméra du mouvement 3D, et c'est donc bien la notion de mouvement apparent qui est apprise. Cependant, si on présentait à une telle méthode des données d'entraînement où des variations dues aux réflexions se superposent à la projection du mouvement 3D, et pour lesquelles la VT est bien la projection du mouvement 3D (donc en utilisant des séquences de TartanAir comme données d'entraînement, pour un *finetuning* par exemple), on pourrait peut-être obtenir une méthode estimant le mouvement 2D correspondant au mouvement 3D des objets, et non le mouvement apparent. Autrement dit, cette méthode apprendrait la notion de réflexion spéculaire et chercherait à ignorer la réflexion et à estimer le mouvement de la surface elle-même. Ceci serait sans doute souhaitable pour une application comme le SLAM, ou tout autre application cherchant à caractériser l'environnement 3D.

6.3.3 Séquence issue de Star Wars IV

Dans la séquence issue de *Star Wars*, la caméra recule et entraîne un mouvement global convergent vers le centre de l'image, avec des déplacements de très grande amplitude proche des bords du champ. Au milieu du champ, un chasseur vole dans la direction de la caméra et présente un mouvement apparent nul, ou du moins très faible par rapport au mouvement global. Les résultats sont présentés sur la figure 6.8. L'allure du mouvement convergent peut être vérifiée en utilisant la légende rappelée sur la figure 6.1. On retrouve la bonne allure globale sur les quatre estimations. Les estimations proposées par IRR-PWC et STaRFlow sont néanmoins nettement plus propres que celles des méthodes classiques qui sont un peu trop constantes par morceau. STaRFlow est la seule des quatre méthodes qui parvient à donner une estimation ayant la bonne allure sur la totalité du

champ image. La forme du chasseur au centre est également mieux restituée.

6.4 Mesure de champ de vitesses pour la métrologie en mécanique

La mesure de champ de vitesses par imagerie implique de traiter des images d'aspect très différent de celles que nous avons traitées ci-dessus, et de celles vues par IRR-PWC et STaRFlow lors de l'entraînement. Nous allons tester ces méthodes sur deux séquences de mesure par imagerie, en mécanique du solide (déformation et rupture d'un matériau sous contrainte) et en mécanique des fluides (mesure du champ de vitesses d'un écoulement par PIV).

6.4.1 Séquence de déformation et rupture d'un matériau

La mesure de champs de déplacements ou de déformations par corrélation d'image est un outil important de l'étude des matériaux. Elle consiste à faire une série d'images d'une éprouvette du matériau que l'on cherche à caractériser alors qu'il subit des contraintes variables via une machine à traction. On estime le champ des déplacements de la surface de l'éprouvette par corrélation d'image (ou par d'autres méthodes d'estimation du mouvement apparent) — et très souvent on dérive ce champ de déplacements pour obtenir le champ des déformations. Pour faciliter et améliorer l'estimation, la surface de l'éprouvette est souvent recouverte d'un mouchetis de peinture fournissant une texture bien contrastée dans les images. L'exemple présenté ici, issu des travaux de thèse de Thomas Corre à l'École Centrale de Nantes (CORRE [2018]) est une séquence dans laquelle une membrane de polyuréthane se déchire de part en part sous l'effet d'une traction effectuée sur ses bords et par l'amorce d'une fissure à une extrémité par une lame [1].

La figure 6.9 présente les résultats de FOLKI, DeepFlow, IRR-PWC et STaRFlow sur cette séquence. Dans la partie texturée, les quatre méthodes proposent des estimations cohérentes. Sur cet exemple les méthodes IRR-PWC et STaRFlow généralisent la tâche de l'estimation du mouvement à des données très différentes de celles vues pendant l'entraînement. STaRFlow présente un résultat qui nous paraît meilleur que celui obtenu avec IRR-PWC, notamment au point central de la cassure (à la frontière entre le jaune et le violet sur la représentation colorée du flot). L'estimation de STaRFlow à cet endroit est plus proche de ce qu'estiment les méthodes classiques, là où IRR-PWC sous-estime l'amplitude du déplacement. STaRFlow offre un bon compromis entre précision et lissage dans cette zone.

La partie homogène blanche donne un comportement qu'il est intéressant d'analyser. Les méthodes par apprentissage profond estiment que cette zone se déplace vers la droite, comme la déchirure. Ces méthodes ont probablement appris à propager le mouvement dans les zones homogènes en considérant qu'elles appartiennent au même corps rigide. On remarque le même phénomène, bien que moins étendu, dans l'estimation de DeepFlow. Dans ce cas, il est probable qu'il s'agisse d'un *inpainting* spatial dû au terme de régularisation, supposant une évolution lente du flot spatialement. Ici, seul le terme de régularisation peut être utilisé, la conservation de l'intensité n'apportant aucune information dans une zone complètement homogène.

1. la séquence des images acquises est mise à disposition par l'École Centrale de Nantes à l'adresse https://zenodo.org/record/4034638#.X5A2NZrgr0p

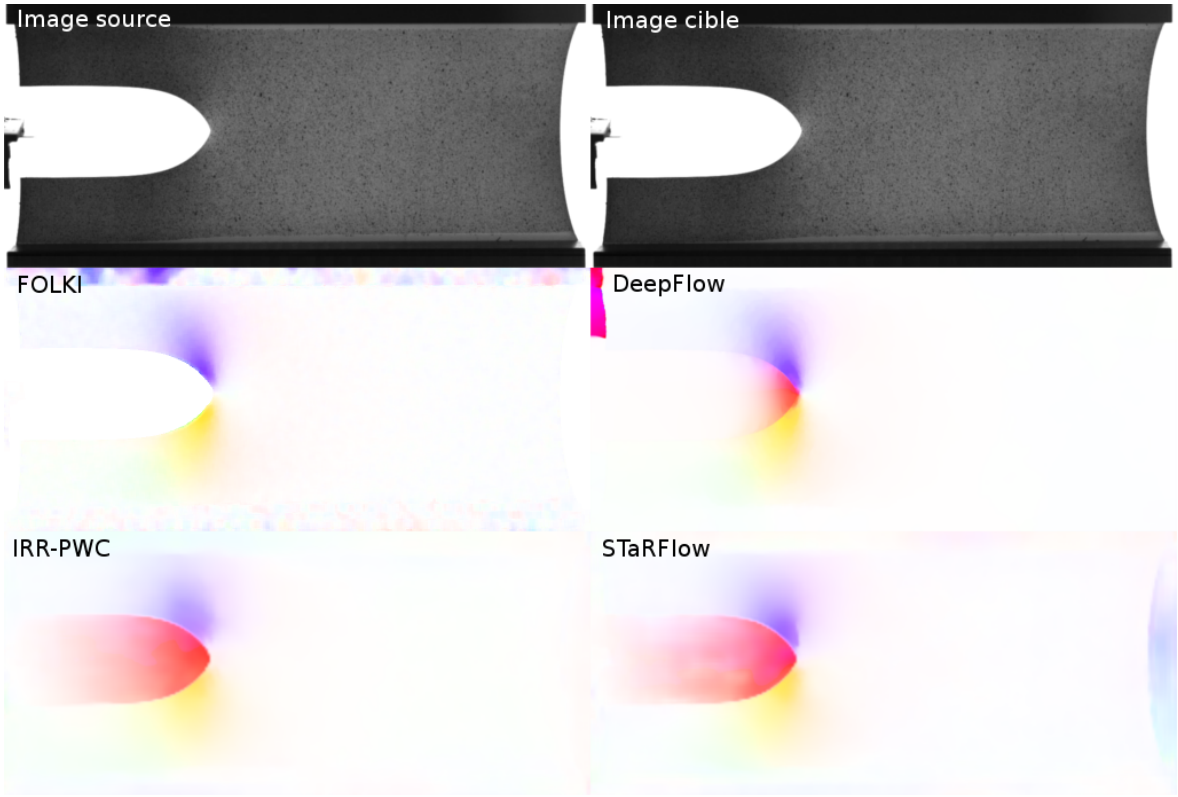

FIGURE 6.9 – Estimation du flot optique sur la séquence de rupture d'un matériau par les méthodes FOLKI (3 niveaux, rayon de la fenêtre = 6 pixels), DeepFlow (sans *matching* et avec le jeu de paramètres par défaut), IRR-PWC et STaRFlow ($N' = 4$).

Si on s'intéresse de près à la frontière entre la zone blanche et la zone texturée, on peut remarquer, sur les estimations de IRR-PWC et STaRFlow, un aspect crénelé qui n'apparaît pas sur l'estimation de DeepFlow. Ceci est probablement dû au fait que l'on estime le flot au quart de la résolution, et passe à pleine résolution par interpolation bilinéaire. À nouveau, il serait dans ce cas intéressant d'explorer la question d'une estimation à pleine résolution dans un algorithme comme STaRFlow.

Par ailleurs, dans ce contexte très différent de celui des données d'entraînement, il paraît intéressant de réentraîner, ou compléter l'entraînement, en utilisant des données dédiées au contexte applicatif d'intérêt. Des travaux ont récemment été mené à ce sujet par BOUKHTACHE et collab. [2020], qui proposent un entraînement (*finetuning*) sur des données synthétiques associées à ce contexte et comparent les résultats obtenus avec plusieurs architectures (notamment FlowNetS, FlowNetC, LiteFlowNet et PWC-Net). Ils proposent ensuite, une variante de FlowNetS pour estimer le mouvement à pleine résolution et montrent justement que cette modification améliore les résultats, ce qui incite à nouveau à explorer l'estimation à pleine résolution sur STaRFlow si l'on cherche à l'adapter à ce type de séquences. [2]

6.4.2 Séquence de l'aile battante (PIV)

Nous reprenons ici la séquence de l'aile battante présentée dans le chapitre 3. Rappelons qu'il s'agit d'une séquence simulée d'images de particules, simulant l'écoulement fluide dans le sillage d'une aile battante. La figure 6.10 montre les résultats obtenus avec LKFT (nous utilisons notre réimplémentation utilisant la forme inverse, présentée dans le chapitre 3), IRR-PWC, et STaRFlow avec deux horizons temporels différents $N' = 4$ et $N' = 5$.

Sur cette séquence, l'algorithme IRR-PWC présente un résultat complètement erroné, sans doute à cause de l'aspect trop différent des images, et au type de mouvement qui n'a rien à voir avec ce qui a pu être appris pendant l'entraînement. Avec $N' = 4$, STaRFlow présente une zone où l'estimation est complètement fausse en bas à droite du champ. L'estimation semble correcte dans la partie gauche. Le problème de cette zone aberrante est résolu en prenant une image supplémentaire dans l'horizon temporel considéré ($N' = 5$). Dans ce cas, bien que le résultat soit trop peu précis par rapport à l'estimation de LKFT et pour les exigences du contexte de la métrologie, l'estimation est étonnamment bonne étant donné le contexte extrêmement différent de celui des données d'entraînement. Ce résultat demeure donc encourageant, et invite à approfondir la question de l'utilisation de ce type de méthode pour la mesure par imagerie.

Pour cette expérience, contrairement à celle réalisée dans le chapitre 3 sur la même séquence, nous n'avons pas ajouté de bruit aux images. Avec le même ajout de bruit que celui réalisé dans le chapitre 3, le problème de STaRFlow constaté avec $N' = 4$ est empiré, et n'est pas résolu avec $N' > 4$. Il y a donc tout de même, et ce n'est pas si étonnant, un problème de généralisation à ce contexte, se traduisant notamment par un côté imprévisible et peu robuste du fonctionnement de l'algorithme. Ici aussi, il paraît préférable d'envisager un entraînement sur des données dédiées. Ceci a été proposé par CAI et collab. [2019]

2. Rappelons que FlowNetS est composé d'un encodeur avec 6 niveaux de sous-échantillonnage (d'un facteur 2 à chaque fois), suivi d'un décodeur qui ne comporte que 4 niveaux. La résolution finale du flot en sortie est donc quatre fois moindre que celle des images d'entrée. On peut augmenter la résolution de sortie en ajoutant des niveaux au décodeur, ou bien en retirant des sous-échantillonnages dans l'encodeur (si on ne sous-échantillonne que 4 fois et qu'on sur-échantillonne ensuite 4 fois on arrive bien à pleine résolution). Les travaux de BOUKHTACHE et collab. [2020] montrent qu'on obtient de meilleurs résultats en retirant des sous-échantillonnages dans l'encodeur, mais pas en ajoutant des niveaux au décodeur.

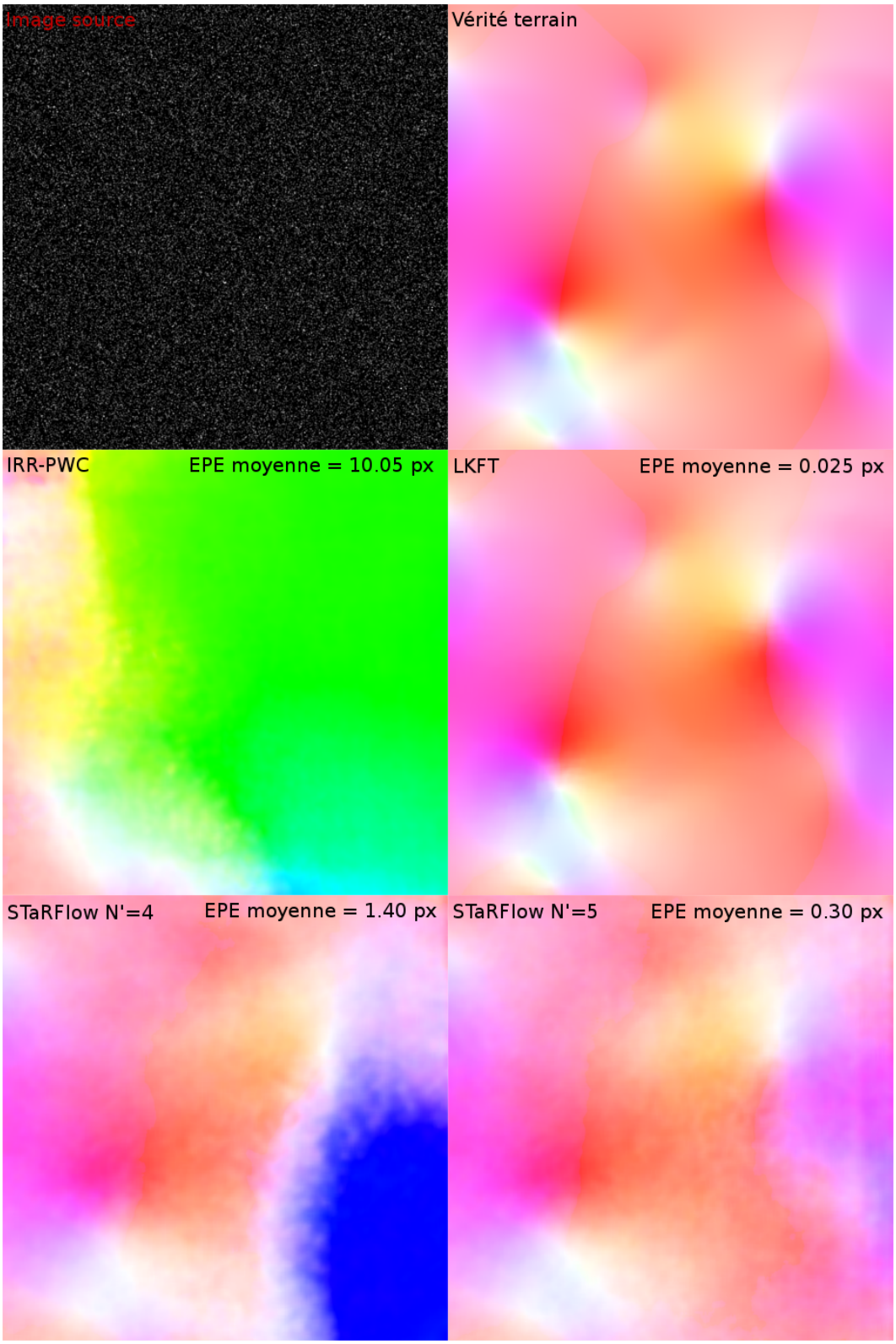

FIGURE 6.10 – Estimation du flot optique sur la séquence de l'aile battante par les méthodes LKFT (forme inverse, 4 niveaux d'échelle, rayon de la fenêtre = 16 pixels, $N = 8$ instants, modèle polynomial de degré $d = 3$), IRR-PWC et STaRFlow.

en utilisant l'architecture LiteFlowNet (HUI et collab. [2018]). Ces travaux donnent des résultats très encourageants.

6.5 Conclusion du chapitre

Nous avons pu constater que les méthodes par apprentissage profond, IRR-PWC et STaRFlow, présentaient de bonnes capacités de généralisation. Sur les données réelles, les résultats sont cohérents avec les estimations données par FOLKI et DeepFlow, et apparaissent en général qualitativement bien meilleurs, alors que les données d'entraînement de STaRFlow et IRR-PWC sont synthétiques. Ces méthodes donnent même des résultats cohérents sur la séquence de déformation et rupture d'un matériau, d'aspect extrêmement différent de celui des données d'entraînement.

Les résultats de STaRFlow sur les séquences étudiées sont particulièrement satisfaisants, montrant notamment une grande robustesse à de mauvaises conditions d'illumination (figure 6.6), une très bonne restitution des détails spatiaux fins (figure 6.7), et une excellente segmentation des objets sur les différentes séquences réelles de section 6.2. Ceci indique un potentiel très intéressant de STaRFlow pour des applications comme :

— la détection d'objets mobiles et la prédiction de leur trajectoire, notamment pour éviter une collision dans le contexte de la navigation (en particulier dans le cas de la séquence du piéton).

— la reconnaissance d'action, grâce à la bonne segmentation des objets et la propreté globale des flots estimés (par exemple sur la séquence de jonglage).

— le SLAM (même en conditions visuelles difficiles, notamment la nuit, comme le montre le résultat obtenu sur la séquence nocturne de TartanAir), vue la propreté des estimations du mouvement global sur les séquences de TartanAir et sur la séquence issue de StarWars.

— toute application multi-image, en permettant un recalage précis et propre des images, même en présence d'objets mobiles. Pour le cas de la super-résolution, on aurait peut-être intérêt à considérer la question d'une estimation de flot optique à pleine résolution.

Cependant, les séquences de la foule dans la gare et de déformation et rupture d'un matériau semblent mettre en évidence une limite de résolution des méthodes IRR-PWC et STaRFlow, probablement due au fait que l'estimation est réalisée au quart de la résolution des images d'entrée avant d'être sur-échantillonnée à pleine résolution par interpolation bilinéaire. Il serait donc intéressant de réfléchir à une manière plus précise de passer à pleine résolution avec ces méthodes. Les travaux de BOUKHTACHE et collab. [2020] abordent cette question, dans le contexte de la mesure par imagerie de déformation des matériaux, autour de l'architecture FlowNetS DOSOVITSKIY et collab. [2015].

Enfin, si la plupart de nos résultats montrent une bonne capacité de généralisation à de nombreux contextes, celui de la mesure par imagerie, et en particulier la PIV, reste particulièrement différent. De plus, ce contexte métrologique implique une certaine précision et compréhension des outils utilisés. Les résultats sur la séquence de l'aile battante montrent que le fonctionnement des méthodes IRR-PWC et STaRFlow dans ce contexte est imprévisible, en l'état de nos connaissances. Cependant, il est encourageant de voir qu'on obtient, dans certains cas comme celui de la figure 6.10 avec $N' = 5$, un résultat correct dans un contexte si particulier. Pour un tel contexte il paraît donc naturel de proposer un entraînement dédié, sur des données de mesure par imagerie.

6.6 Références

BOUKHTACHE, S., K. ABDELOUAHAB, F. BERRY, B. BLAYSAT, M. GREDIAC et F. SUR. 2020, «When deep learning meets digital image correlation», *Optics and Lasers in Engineering*, vol. 136, p. 106 308. 153, 155

BROX, T., A. BRUHN, N. PAPENBERG et J. WEICKERT. 2004, «High accuracy optical flow estimation based on a theory for warping», *Computer Vision-ECCV 2004*, p. 25–36. 139

CAESAR, H., V. BANKITI, A. H. LANG, S. VORA, V. E. LIONG, Q. XU, A. KRISHNAN, Y. PAN, G. BALDAN et O. BEIJBOM. 2019, «nuScenes : A multimodal dataset for autonomous driving», *arXiv preprint arXiv :1903.11027*. 141

CAI, S., J. LIANG, Q. GAO, C. XU et R. WEI. 2019, «Particle image velocimetry based on a deep learning motion estimator», *IEEE Transactions on Instrumentation and Measurement*, vol. 69, n° 6, p. 3538–3554. 153

CORRE, T. 2018, *Rupture dynamique de membranes élastomères : étude expérimentale par mesure de champs*, thèse de doctorat. Thèse de doctorat dirigée par Verron, Erwan Mécanique des solides, des matériaux, des structures et des surfaces Ecole centrale de Nantes 2018. 151

DOSOVITSKIY, A., P. FISCHER, E. ILG, P. HAUSSER, C. HAZIRBAS, V. GOLKOV, P. VAN DER SMAGT, D. CREMERS et T. BROX. 2015, «FlowNet : Learning optical flow with convolutional networks», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 2758–2766. 155

HUI, T.-W., X. TANG et C. C. LOY. 2018, «LiteFlowNet : A lightweight convolutional neural network for optical flow estimation», dans *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 155

HUR, J. et S. ROTH. 2019, «Iterative residual refinement for joint optical flow and occlusion estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5754–5763. 138

LE BESNERAIS, G. et F. CHAMPAGNAT. 2005, «Dense optical flow by iterative local window registration», dans *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, IEEE, p. I–137. 138

WANG, W., D. ZHU, X. WANG, Y. HU, Y. QIU, C. WANG, Y. HU, A. KAPOOR et S. SCHERER. 2020, «Tartanair : A dataset to push the limits of visual slam», . 147

WEINZAEPFEL, P., J. REVAUD, Z. HARCHAOUI et C. SCHMID. 2013, «Deepflow : Large displacement optical flow with deep matching», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 1385–1392. 138

Chapitre 7

Conclusion et perspectives

Sommaire

7.1 Comment exploiter la cohérence temporelle ?

Nous revenons ici sur les différentes méthodes considérées pour tenir compte d'une information temporelle sur plusieurs instants consécutifs. Nous avons étudié des approches considérant un voisinage temporel de taille fixe et des méthodes récurrentes, nous avons également distingué deux formalismes en considérant soit des trajectoires soit des déplacements instantanés.

7.1.1 Apprentissage et utilisation d'une base de modèles temporels

Nous avons développé, dans le chapitre 3, l'algorithme FOLKI-MF permettant d'estimer les coefficients de la décomposition du mouvement sur une base de modèles temporels arbitraires. Nous avons également proposé d'extraire une base à partir des données d'intérêt par analyse en composantes principales sur un sous-ensemble de trajectoires. Nous avons ainsi montré, dans le contexte de la mesure par imagerie de champs de vitesses d'écoulements fluides, une réduction du biais de mesure, par rapport à une modélisation polynomiale de l'évolution temporelle, tout en continuant de bénéficier de la réduction de variance offerte par l'exploitation de la continuité temporelle.

Cependant, nous apprenons une base commune à tout le champ spatial, et avons remarqué dans nos expériences qu'elle n'était pas adaptée partout. On comprend bien que l'évolution temporelle peut être très différente dans des endroits différents du champ. Il serait intéressant de réaliser une segmentation spatiale et d'estimer une base différente par région. On pourrait ainsi obtenir des modèles temporels localement adaptés.

Dans nos expériences, nous avons appris la base à partir des trajectoires utilisées pour générer la séquence d'images, ce qui n'est pas réalisable sur une séquence réelle. Notre travail constitue donc, à ce stade, une preuve de concept. Il faudrait maintenant étudier l'apprentissage de la base à partir de trajectoires issues d'un suivi de points ou d'une simulation associée. Il faudrait également réaliser une évaluation sur d'autres séquences, et généraliser à des séquences expérimentales réelles. Pour mettre en évidence l'intérêt d'une base apprise, il faudra trouver des séquences avec un niveau suffisant de complexité, sinon le modèle adapté sera polynomial. Une adaptation de l'algorithme pour la mesure de champs de vitesses 3D pourrait être envisagée, pour la mesure de champs de déplacements 3D sur des échantillons volumiques reconstruits par tomographie.

7.1.2 Gestion de l'horizon temporel

Nous avons considéré plusieurs méthodes pour exploiter un horizon temporel de plus de 2 images. FOLKI-MF (chapitre 3) et les architectures "Stack" (chapitre 4) considèrent une fenêtre temporelle de taille fixe autour d'un instant de référence, tandis que STaR-Flow (chapitre 5) est une méthode récurrente considérant deux instants à la fois mais en gardant une mémoire du passé. Dans le cas pratique d'une estimation de mouvement au fur et à mesure que de nouvelles images sont acquises, considérer une fenêtre temporelle autour de l'instant d'intérêt implique un léger retard (correspondant à la moitié de la largeur temporelle de la fenêtre) de l'estimation. Le fonctionnement récurrent au contraire donne directement une estimation au dernier instant. Cependant, la première estimation n'exploite, dans ce cas, aucune continuité temporelle. Nous avons obtenu de meilleurs résultats avec le fonctionnement récurrent, mais peut-être que l'horizon temporel considéré dans nos méthodes "Stack" était trop long pour les séquences étudiées, rendant donc l'estimation plus complexe. On pourrait donc réduire le nombre d'images

pour voir si cela améliore les résultats. La tâche est un peu laborieuse dans la mesure où il faut réentraîner une méthode spécifique à chaque nombre d'images. Une idée que nous n'avons pas expérimentée est celle d'utiliser des convolutions 3D. Au lieu de présenter au réseau un tenseur dont les canaux représentent les instants, on pourrait ajouter une dimension pour l'axe temporel et utiliser des convolutions 3D, permettant un horizon temporel variable. Ces techniques de convolution 3D sont utilisées fréquemment dans le domaine de la reconnaissance d'actions (TRAN et collab. [2015, 2018]).

Nous avons également distingué l'estimation dense de trajectoires de l'estimation d'une séquence de champs de déplacements instantanés. La modélisation temporelle utilisée par FOLKI-MF caractérise la trajectoire d'un élément de fluide. Cette représentation présente l'avantage de donner accès directement à des portions de trajectoires, en chaque pixel d'une image de référence. Une telle représentation pourrait être considérée dans le cas d'un recalage de $N > 2$ images dans une même géométrie. Cependant dans un cadre général, si on souhaite agrandir l'horizon temporel, cette représentation pose le problème de l'amplitude croissante des mouvements et, pire, des sorties du champ de la caméra. D'autre part, les outils disponibles dans la communauté du flot optique, notamment les annotations des jeux de données, sont basés sur une représentation en champs de déplacements instantanés. Nous avons, dans le chapitre 4, implémenté et comparé les deux représentations dans une approche basée sur l'apprentissage profond. La représentation en champs instantanés permet de réaliser une phase d'apprentissage sur FlyingThings3D et de construire une architecture basée sur PWC-Net, ce qui permet d'obtenir de meilleurs résultats. Enfin, STaRFlow (chapitre 5) estime les champs de déplacements instantanés, bien que le recalage des informations du passé dans la géométrie courante rappelle la représentation en trajectoires.

7.1.3 Exploitation d'une mémoire du passé

Dans le chapitre 5, nous avons développé STaRFlow, un algorithme basé sur l'apprentissage profond tirant profit d'informations mémorisées du passé pour améliorer l'estimation courante, dans un fonctionnement récurrent. Ce fonctionnement récurrent correspond à un scénario réel dans lequel l'algorithme voit défiler une séquence d'images et donne pour chaque nouvelle paire une estimation du flot optique. Notre méthode présente des résultats à l'état de l'art sur les *benchmarks* MPI Sintel et KITTI 2015, tout en étant nettement plus rapide que les meilleures méthodes variationnelles et plus légère, en termes de nombre de paramètres, que les méthodes concurrentes utilisant également des réseaux de neurones convolutifs.

Alors que la plupart des méthodes actuelles se limitent à 3 instants, nous avons montré que la récurrence temporelle implémentée dans STaRFlow permettait de profiter de la continuité temporelle pour améliorer les résultats sur un horizon temporel atteignant 5 instants (section 5.3.2), et ce, bien que 4 instants seulement ont été considérés lors de l'entraînement.

7.2 Quel est l'intérêt d'une estimation multiframe ?

Nous synthétisons ici les cas dans lesquels nous avons observé une amélioration grâce à l'exploitation de la continuité temporelle, en considérant $N > 3$ instants. Comme nous l'avons vu à la fin du chapitre 5 il existe aussi des cas où l'utilisation de l'information temporelle peut dégrader l'estimation, notamment dans le cas de discontinuités temporels du mouvement. Ceci reste peu fréquent et n'est pas significativement pénalisant en

moyenne, comme le montrent nos résultats sur Sintel par exemple. Dans la suite de cette section on se focalise donc sur les bénéfices apportés par le multiframe.

7.2.1 Dégradations de la qualité image

Considérer des instants supplémentaires permet de gagner en robustesse aux dégradations de la qualité image. Nous avons vu dans le chapitre 3, l'intérêt du multiframe dans le cas de données bruitées, avec LKFT et sa dépendance polynomiale comme avec FOLKI-MF utilisant une modélisation apprise du mouvement. Dans le chapitre 4 la supériorité de notre méthode multiframe FlowNetStack sur les méthodes biframe est plus marquée dans le cas des données de ChairsMultiframe "final" incluant du bruit additif gaussien et du flou gaussien asymétrique. Enfin, nous avons montré au chapitre 5 l'intérêt de notre connexion temporelle dans le cas des données de Sintel Final —incluant du flou de défocalisation, du flou de bougé et des effets de brouillard — et dans le chapitre 6 son intérêt (en comparant STaRFlow à IRR-PWC sur la figure 6.6) dans des conditions de faible illumination [1].

7.2.2 Finesse spatiale

La régularisation temporelle offerte par l'agrandissement de l'horizon temporel permet de relâcher la régularisation spatiale. Par exemple, dans une méthode à fenêtre comme FOLKI, la taille de fenêtre optimale du point de vue de l'erreur de mesure est plus faible dans le cas multiframe, comme on peut le constater sur la figure 3.10. On peut ainsi obtenir une meilleure résolution spatiale, et bénéficier d'une meilleure estimation des structures spatiales fines. C'est ce que nous avons d'ailleurs observé dans nos expériences avec FlowNetStack sur ChairsMultiframeVal en section 4.2.4. Dans le chapitre 5, STaRFlow parvient à estimer le mouvement de petits objets qui échappent à la méthode biframe IRR-PWC (la principale différence entre les deux méthodes est l'ajout de la récurrence temporelle). Enfin, les champs de mouvement estimés par STaRFlow présentent visuellement une meilleure segmentation des objets, comme nous avons pu l'observer dans les chapitres 5 et 6.

7.2.3 Occultations

Notre méthode STaRFlow donne de très bons résultats dans les zones d'occultations, c'est-à-dire qui sont occultées dans l'image cible. La mémoire du passé permet de réaliser un *inpainting* temporel dans ces zones, en extrapolant le mouvement à partir des estimations précédentes. Nous l'avons montré dans le chapitre 5, et également observé dans le chapitre 6 avec l'exemple de la figure 6.4.

7.2.4 Petits mouvements

L'estimation des mouvements de faibles amplitudes est *a priori* un cas sur lequel on s'attend à obtenir une amélioration grâce au multiframe. Nous l'avons effectivement constatée dans le chapitre 4. Nos expériences n'ont pas mis en évidence un apport de STaRFlow sur l'estimation des petits mouvements. De fait, l'algorithme STaRFlow a été

1. Dans l'application de STaRFlow à la PIV dans le chapitre 6, nous avons constaté que l'algorithme était mis en défaut sur des données bruitées mais cela relève à ce stade, à notre avis, d'un problème de généralisation plus que d'un manque de robustesse au bruit.

conçu et entraîné, dans le but d'être performant sur les *benchmarks* Sintel et KITTI dans lesquels les petits mouvements ne sont pas l'enjeu principal. Il est cependant probable que cette architecture puisse permettre l'amélioration de l'estimation de ces mouvements, moyennant un éventuel réentraînement.

7.3 Utilisation du flot optique en pratique

Interrogeons-nous, enfin, sur l'utilisabilité des méthodes de flot optique en pratique. Celle-ci dépend d'un compromis entre qualité de l'estimation et temps de calcul. La rapidité de l'algorithme est capitale dans de nombreux domaines applicatifs :

— dans un contexte embarqué, pour une application dans le domaine de la robotique par exemple, les moyens de calculs sont limités et il est en général nécessaire de proposer un fonctionnement en temps réel.

— dans le domaine de la mesure et en recalage, il est fréquent d'avoir un volume important de données à traiter et l'utilisateur apprécie de pouvoir disposer des résultats rapidement (sans attendre une nuit de traitement).

Malgré leurs résultats de très bonne qualité, les méthodes variationnelles sont trop lentes pour ces contextes, avec des temps de calcul atteignant plusieurs secondes par paire d'images. Les méthodes par apprentissage ont révolutionné ce compromis qualité/performance : elles sont à la fois plus précises que les méthodes variationnelles, établissant le nouvel état de l'art sur des *benchmarks* comme Sintel et Kitti, et beaucoup plus rapides avec des temps de traitement de l'ordre du dixième de seconde.

Ces méthodes par apprentissage sont cependant particulièrement lourdes en mémoire, à cause d'un nombre de paramètres atteignant plusieurs millions. Une perspective future pourrait être de réduire le nombre de paramètres d'une méthode comme STaRFlow. Pour cela, on pourrait réétudier l'impact des différents sous-modules de l'architecture, devenus nombreux et peut-être redondants. Cette complexité est en partie due au fait que STaRFlow, comme de nombreuses autres méthodes concurrentes, construit sur l'architecture PWC-Net en y adjoignant de nouveaux modules. L'arrivée de nouvelles architectures, comme celle de RAFT (TEED et DENG [2020]) qui vient de se classer en tête sur Sintel, pourrait conduire à des solutions plus légères à l'avenir.

Par ailleurs, les méthodes d'estimation de flot optique basées sur l'apprentissage profond sont encore très récentes et on manque de recul quant à leur bon fonctionnement en pratique. À notre connaissance, peu d'expériences ont été réalisées quant à la capacité de généralisation ou l'explicabilité du fonctionnement des méthodes de flot optique par apprentissage profond. Sur un plan empirique, dans le chapitre 6, nous avons testé sur quelques exemples issus de contextes applicatifs variés la capacité de généralisation de notre meilleure méthode, STaRFlow. Ces quelques évaluations qui, rappelons-le se fondent uniquement sur des modèles appris sur données de synthèse, nous paraissent très encourageantes. Cependant nous avons pu observer des zones où StarFlow est en échec, et nous ne savons pour l'instant pas expliquer le comportement du réseau dans ces cas pathologiques. Cela incite, dans les contextes où la qualité de l'estimation est critique, comme dans la mesure par imagerie, à entraîner les réseaux sur des données issues du contexte applicatif. Se posent néanmoins les questions suivantes : est-il préférable de repartir d'un réseau préentraîné dans un autre contexte et de réaliser un *finetuning*, ou bien de réentraîner le réseau à partir d'une initialisation aléatoire directement sur les données du contexte d'intérêt ? Avec un entraînement sur des données simulées de mesure par imagerie, la méthode généralise-t-elle à des données réelles ?

Enfin, notons que les méthodes basées sur l'utilisation d'un réseau de neurones n'ont pas de paramètre réglable. Ceci peut être vu à la fois comme une force et une faiblesse. Il est vrai que si l'algorithme donne un résultat aberrant, il est impossible de relancer l'estimation avec un autre réglage. Cependant nous n'avons détecté que peu de cas pathologiques dans nos évaluations, et serions donc tentés de voir cette absence de paramètre réglable comme une force, facilitant la mise en œvre et accélérant significativement la prise en main de la méthode pour une utilisation pratique.

7.4 Conclusion

STaRFlow est la méthode la plus performance obtenue durant nos travaux pour estimer le flot optique en profitant de la continuité temporelle. Il s'agit d'une méthode basée sur un réseau de neurone profond entraîné de manière supervisée sur des données synthétiques, exploitant l'information du passé par un fonctionnement récurrent permettant de garder une mémoire des instants précédents. Nous avons montré l'étonnante capacité de généralisation de cette méthode à des données réelles et parfois très différentes des données d'entraînement. Nous avons également montré l'intérêt de l'estimation multi-frame, notamment pour l'estimation du mouvement de petits objets, dans le cas d'occultations entre objets ou encore lorsque la qualité image est dégradée.

Dans le contexte de la mesure, les méthodes classiques locales ou globales restent pour l'instant meilleures que STaRFlow. Une perspective à ce sujet serait d'entraîner STaRFlow — ou une architecture dérivée de cet algorithme — sur des données spécifiques à ce contexte, puis d'évaluer minutieusement sa précision et sa robustesse. D'autre part, nous avons évoqué dans le chapitre 2 l'existence de travaux cherchant à apprendre à estimer le flot optique de manière non supervisée sur des données non annotées. Plutôt que d'entraîner sur des données de simulation uniquement on pourrait aussi imaginer un apprentissage non supervisé sur des données réelles. Il est assez probable que l'architecture STaRFlow, grâce à sa mémoire temporelle, puisse donner des résultats intéressants dans le contexte de la mesure où les données présentent souvent une bonne continuité temporelle. Une autre perspective, est de traiter la question de l'estimation à pleine résolution, et non au quart de la résolution d'entrée comme c'est le cas avec STaRFlow et les méthodes dont STaRFlow s'inspire. Cette limite de résolution est probablement plus limitante dans le contexte de la mesure que dans celui de la vision où d'autres problèmes dominent (comme les grands mouvements et les occultations).

Enfin, si les méthodes par apprentissage profond sont très rapides par rapport à l'état de l'art précédent, elles demeurent très lourdes en mémoire. Une autre perspective est donc de chercher à alléger les architectures pour des résultats équivalents.

7.5 Références

TEED, Z. et J. DENG. 2020, «Raft : Recurrent all-pairs field transforms for optical flow», *arXiv preprint arXiv :2003.12039*. 161

TRAN, D., L. BOURDEV, R. FERGUS, L. TORRESANI et M. PALURI. 2015, «Learning spatiotemporal features with 3d convolutional networks», dans *Proceedings of the IEEE international conference on computer vision*, p. 4489–4497. 159

TRAN, D., H. WANG, L. TORRESANI, J. RAY, Y. LECUN et M. PALURI. 2018, «A closer look at spatiotemporal convolutions for action recognition», dans *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, p. 6450–6459. 159

Chapitre 8

Liste complète des références

BAILER, C., B. TAETZ et D. STRICKER. 2015, «Flow Fields : Dense correspondence fields for highly accurate large displacement optical flow estimation», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 4015–4023.

BAKER, S. et I. MATTHEWS. 2004, «Lucas-Kanade 20 years on : A unifying framework», *International journal of computer vision*, vol. 56, nº 3, p. 221–255.

BAKER, S., D. SCHARSTEIN, J. LEWIS, S. ROTH, M. J. BLACK et R. SZELISKI. 2007, «A database and evaluation methodology for optical flow», *International Conference on Computer Vision*.

BALLAS, N., L. YAO, C. PAL et A. COURVILLE. 2015, «Delving deeper into convolutional networks for learning video representations», *arXiv preprint arXiv :1511.06432*.

BAR-HAIM, A. et L. WOLF. 2020, «ScopeFlow : Dynamic scene scoping for optical flow», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 7998–8007.

BENGIO, Y., J. LOURADOUR, R. COLLOBERT et J. WESTON. 2009, «Curriculum learning», dans *Proceedings of the 26th annual international conference on machine learning*, p. 41–48.

BLACK, M. J. et P. ANANDAN. 1993, «A framework for the robust estimation of optical flow», dans *1993 (4th) International Conference on Computer Vision*, IEEE, p. 231–236.

BOUGUET, J.-Y. 2001, «Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm», *Intel Corporation*, vol. 5, nº 1-10, p. 4.

BOUKHTACHE, S., K. ABDELOUAHAB, F. BERRY, B. BLAYSAT, M. GREDIAC et F. SUR. 2020, «When deep learning meets digital image correlation», *Optics and Lasers in Engineering*, vol. 136, p. 106 308.

BRIGOT, G., E. COLIN-KOENIGUER, A. PLYER et F. JANEZ. 2016, «Adaptation and evaluation of an optical flow method applied to coregistration of forest remote sensing images», *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, nº 7, p. 2923–2939.

BROX, T., C. BREGLER et J. MALIK. 2009, «Large displacement optical flow», dans *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, p. 41–48.

BROX, T., A. BRUHN, N. PAPENBERG et J. WEICKERT. 2004, «High accuracy optical flow estimation based on a theory for warping», *Computer Vision-ECCV 2004*, p. 25–36.

BUTLER, D. J., J. WULFF, G. B. STANLEY et M. J. BLACK. 2012, «A naturalistic open source movie for optical flow evaluation», dans *European Conference on Computer Vision*, Springer, p. 611–625.

CAESAR, H., V. BANKITI, A. H. LANG, S. VORA, V. E. LIONG, Q. XU, A. KRISHNAN, Y. PAN, G. BALDAN et O. BEIJBOM. 2019, «nuScenes : A multimodal dataset for autonomous driving», *arXiv preprint arXiv :1903.11027*.

CAI, S., J. LIANG, Q. GAO, C. XU et R. WEI. 2019, «Particle image velocimetry based on a deep learning motion estimator», *IEEE Transactions on Instrumentation and Measurement*, vol. 69, nº 6, p. 3538–3554.

CHAMPAGNAT, F., A. PLYER, G. LE BESNERAIS, B. LECLAIRE, S. DAVOUST et Y. LE SANT. 2011, «Fast and accurate PIV computation using highly parallel iterative correlation maximization», *Experiments in fluids*, vol. 50, nº 4, p. 1169.

CHANG, A. X., T. FUNKHOUSER, L. GUIBAS, P. HANRAHAN, Q. HUANG, Z. LI, S. SAVARESE, M. SAVVA, S. SONG, H. SU et collab.. 2015, «Shapenet : An information-rich 3D model repository», *arXiv preprint arXiv :1512.03012*.

CHERRIER, N., T. CASTAINGS et A. BOULCH. 2017, «Deep sequence-to-sequence neural networks for ionospheric activity map prediction», dans *International Conference on Neural Information Processing*, Springer, p. 545–555.

CORDTS, M., M. OMRAN, S. RAMOS, T. REHFELD, M. ENZWEILER, R. BENENSON, U. FRANKE, S. ROTH et B. SCHIELE. 2016, «The cityscapes dataset for semantic urban scene understanding», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 3213–3223.

CORRE, T. 2018, *Rupture dynamique de membranes élastomères : étude expérimentale par mesure de champs*, thèse de doctorat. Thèse de doctorat dirigée par Verron, Erwan Mécanique des solides, des matériaux, des structures et des surfaces Ecole centrale de Nantes 2018.

DOSOVITSKIY, A., P. FISCHER, E. ILG, P. HAUSSER, C. HAZIRBAS, V. GOLKOV, P. VAN DER SMAGT, D. CREMERS et T. BROX. 2015, «FlowNet : Learning optical flow with convolutional networks», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 2758–2766.

EIGEN, D., C. PUHRSCH et R. FERGUS. 2014, «Depth map prediction from a single image using a multi-scale deep network», dans *Advances in neural information processing systems*, p. 2366–2374.

GARG, R., L. PIZARRO, D. RUECKERT et L. AGAPITO. 2010, «Dense multi-frame optic flow for non-rigid objects using subspace constraints», dans *Asian Conference on Computer Vision*, Springer, p. 460–473.

GEIGER, A., P. LENZ et R. URTASUN. 2012, «Are we ready for autonomous driving? the KITTI vision benchmark suite», dans *Conference on Computer Vision and Pattern Recognition (CVPR)*.

GODET, P., A. BOULCH, A. PLYER et G. LE BESNERAIS. 2020, «Starflow : A spatiotemporal recurrent cell for lightweight multi-frame optical flow estimation», dans *25th International Conference on Pattern Recognition, ICPR 2020, Milan, Italy, 10-15 janvier 2021*, IEEE Computer Society.

GODET, P., F. CHAMPAGNAT, G. LE BESNERAIS et A. PLYER. 2019a, «Learning fluid trajectory models for time-resolved PIV», dans *The 13th International Symposium on Particle Image Velocimetry (ISPIV 2019)*.

GODET, P., A. PLYER, A. BOULCH et G. LE BESNERAIS. 2019b, «Estimation de flot optique multiframe par apprentissage profond», dans *XXVIIème Colloque francophone de traitement du signal et des images (GRETSI 2019)*.

GOODFELLOW, I., Y. BENGIO et A. COURVILLE. 2016, *Deep Learning*, MIT Press. http://www.deeplearningbook.org.

HORN, B. K. et B. G. SCHUNCK. 1981, «Determining optical flow», *Artificial intelligence*, vol. 17, nº 1-3, p. 185–203.

HU, P., G. WANG et Y.-P. TAN. 2018, «Recurrent spatial pyramid CNN for optical flow estimation», *IEEE Transactions on Multimedia*, vol. 20, nº 10, p. 2814–2823.

HUANG, G., Z. LIU, K. Q. WEINBERGER et L. VAN DER MAATEN. 2017, «Densely connected convolutional networks», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, p. 3.

HUI, T.-W., X. TANG et C. C. LOY. 2018, «LiteFlowNet : A lightweight convolutional neural network for optical flow estimation», dans *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

HUR, J. et S. ROTH. 2019, «Iterative residual refinement for joint optical flow and occlusion estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5754–5763.

ILG, E., N. MAYER, T. SAIKIA, M. KEUPER, A. DOSOVITSKIY et T. BROX. 2017, «FlowNet 2.0 : Evolution of optical flow estimation with deep networks», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 6.

ILG, E., T. SAIKIA, M. KEUPER et T. BROX. 2018, «Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation», dans *European Conference on Computer Vision*, Springer, p. 626–643.

IRANI, M. 2002, «Multi-frame correspondence estimation using subspace constraints», *International Journal of Computer Vision*, vol. 48, nº 3, p. 173–194.

JALLAS, D., O. MARQUET et D. FABRE. 2017, «Linear and nonlinear perturbation analysis of the symmetry breaking in time-periodic propulsive wakes», *Physical Review E*, vol. 95, nº 6, p. 063 111.

JANAI, J., F. GUNEY, A. RANJAN, M. BLACK et A. GEIGER. 2018, «Unsupervised learning of multi-frame optical flow with occlusions», dans *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 690–706.

JASON, J. Y., A. W. HARLEY et K. G. DERPANIS. 2016, «Back to basics : Unsupervised learning of optical flow via brightness constancy and motion smoothness», dans *European Conference on Computer Vision*, Springer, p. 3–10.

JEON, Y. J., L. CHATELLIER et L. DAVID. 2014, «Fluid trajectory evaluation based on an ensemble-averaged cross-correlation in time-resolved PIV», *Experiments in fluids*, vol. 55, nº 7, p. 1766.

KENNEDY, R. et C. J. TAYLOR. 2015, «Optical flow with geometric occlusion estimation and fusion of multiple frames», dans *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, p. 364–377.

KINGMA, D. P. et J. BA. 2014, «Adam : A method for stochastic optimization», *arXiv preprint arXiv :1412.6980*.

KRIZHEVSKY, A., I. SUTSKEVER et G. E. HINTON. 2012, «Imagenet classification with deep convolutional neural networks», dans *Advances in neural information processing systems*, p. 1097–1105.

LE BESNERAIS, G. et F. CHAMPAGNAT. 2005, «Dense optical flow by iterative local window registration», dans *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, IEEE, p. I–137.

LIU, L., J. ZHANG, R. HE, Y. LIU, Y. WANG, Y. TAI, D. LUO, C. WANG, J. LI et F. HUANG. 2020, «Learning by analogy : Reliable supervision from transformations for unsupervised optical flow estimation», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 6489–6498.

LIU, P., M. R. LYU, I. KING et J. XU. 2019, «SelFlow : Self-supervised learning of optical flow», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

LUCAS, B. D., T. KANADE et collab.. 1981, «An iterative image registration technique with an application to stereo vision», .

LYNCH, K. et F. SCARANO. 2013, «A high-order time-accurate interrogation method for time-resolved PIV», *Measurement Science and Technology*, vol. 24, nº 3, p. 035 305.

MAURER, D. et A. BRUHN. 2018, «ProFlow : Learning to predict optical flow», dans *Proceedings of the British Machine Vision Conference*.

MAYER, N., E. ILG, P. FISCHER, C. HAZIRBAS, D. CREMERS, A. DOSOVITSKIY et T. BROX. 2018, «What makes good synthetic training data for learning disparity and optical flow estimation?», *International Journal of Computer Vision*, p. 1–19.

MAYER, N., E. ILG, P. HAUSSER, P. FISCHER, D. CREMERS, A. DOSOVITSKIY et T. BROX. 2016, «A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4040–4048.

MEISTER, S., J. HUR et S. ROTH. 2018, «UnFlow : Unsupervised learning of optical flow with a bidirectional census loss», dans *Thirty-Second AAAI Conference on Artificial Intelligence*.

MENZE, M. et A. GEIGER. 2015, «Object scene flow for autonomous vehicles», dans *Conference on Computer Vision and Pattern Recognition*.

NEORAL, M., J. ŠOCHMAN et J. MATAS. 2018, «Continual occlusion and optical flow estimation», dans *Asian Conference on Computer Vision*, Springer, p. 159–174.

PLYER, A., G. LE BESNERAIS et F. CHAMPAGNAT. 2016, «Massively parallel Lucas Kanade optical flow for real-time video processing applications», *Journal of Real-Time Image Processing*, vol. 11, nº 4, p. 713–730.

RANJAN, A. et M. J. BLACK. 2017, «Optical flow estimation using a spatial pyramid network», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2.

REN, Z., O. GALLO, D. SUN, M.-H. YANG, E. SUDDERTH et J. KAUTZ. 2019, «A fusion approach for multi-frame optical flow estimation», dans *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, p. 2077–2086.

REN, Z., J. YAN, B. NI, B. LIU, X. YANG et H. ZHA. 2017, «Unsupervised deep learning for optical flow estimation», dans *Thirty-First AAAI Conference on Artificial Intelligence*.

REVAUD, J., P. WEINZAEPFEL, Z. HARCHAOUI et C. SCHMID. 2015, «Epicflow : Edge-preserving interpolation of correspondences for optical flow», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 1164–1172.

REVAUD, J., P. WEINZAEPFEL, Z. HARCHAOUI et C. SCHMID. 2016, «Deepmatching : Hierarchical deformable dense matching», *International Journal of Computer Vision*, vol. 120, nº 3, p. 300–323.

RONNEBERGER, O., P. FISCHER et T. BROX. 2015, «U-Net : Convolutional networks for biomedical image segmentation», dans *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, p. 234–241.

ROS, G., L. SELLART, J. MATERZYNSKA, D. VAZQUEZ et A. M. LOPEZ. 2016, «The synthia dataset : A large collection of synthetic images for semantic segmentation of urban scenes», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 3234–3243.

SHI, J. et collab.. 1994, «Good features to track», dans *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, IEEE, p. 593–600.

SIMONYAN, K. et A. ZISSERMAN. 2014, «Very deep convolutional networks for large-scale image recognition», *arXiv preprint arXiv :1409.1556*.

STANISLAS, M., K. OKAMOTO, C. J. KÄHLER, J. WESTERWEEL et F. SCARANO. 2008, «Main results of the third international PIV challenge», *Experiments in Fluids*, vol. 45, nº 1, p. 27–71.

SUN, D., S. ROTH et M. J. BLACK. 2014, «A quantitative analysis of current practices in optical flow estimation and the principles behind them», *International Journal of Computer Vision*, vol. 106, nº 2, p. 115–137.

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018a, «Models matter, so does training : An empirical study of CNNs for optical flow estimation», *arXiv preprint arXiv :1809.05571*.

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018b, «PWC-Net : CNNs for optical flow using pyramid, warping, and cost volume», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 8934–8943.

TAO, X., H. GAO, R. LIAO, J. WANG et J. JIA. 2017, «Detail-revealing deep video super-resolution», dans *The IEEE International Conference on Computer Vision (ICCV)*.

TEED, Z. et J. DENG. 2020, «Raft : Recurrent all-pairs field transforms for optical flow», *arXiv preprint arXiv :2003.12039*.

THIELICKE, W. et E. STAMHUIS. 2014, «PIVlab–towards user-friendly, affordable and accurate digital particle image velocimetry in matlab», *Journal of open research software*, vol. 2, nº 1.

TOMASI, C. et T. KANADE. 1992, «Shape and motion from image streams under orthography : a factorization method», *International Journal of Computer Vision*, vol. 9, nº 2, p. 137–154.

TRAN, D., L. BOURDEV, R. FERGUS, L. TORRESANI et M. PALURI. 2015, «Learning spatiotemporal features with 3d convolutional networks», dans *Proceedings of the IEEE international conference on computer vision*, p. 4489–4497.

TRAN, D., H. WANG, L. TORRESANI, J. RAY, Y. LECUN et M. PALURI. 2018, «A closer look at spatiotemporal convolutions for action recognition», dans *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, p. 6450–6459.

UMMENHOFER, B., H. ZHOU, J. UHRIG, N. MAYER, E. ILG, A. DOSOVITSKIY et T. BROX. 2017, «Demon : Depth and motion network for learning monocular stereo», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5038–5047.

VOLZ, S., A. BRUHN, L. VALGAERTS et H. ZIMMER. 2011, «Modeling temporal coherence for optical flow», dans *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, p. 1116–1123.

WANG, C.-M., K.-C. FAN, C.-T. WANG et collab.. 2008, «Estimating optical flow by integrating multi-frame information», *Journal of Information Science and Engineering*, vol. 24, nº 6, p. 1719–1731.

WANG, W., D. ZHU, X. WANG, Y. HU, Y. QIU, C. WANG, Y. HU, A. KAPOOR et S. SCHERER. 2020, «Tartanair : A dataset to push the limits of visual slam», .

WEINZAEPFEL, P., J. REVAUD, Z. HARCHAOUI et C. SCHMID. 2013, «Deepflow : Large displacement optical flow with deep matching», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 1385–1392.

XINGJIAN, S., Z. CHEN, H. WANG, D.-Y. YEUNG, W.-K. WONG et W.-C. WOO. 2015, «Convolutional lstm network : A machine learning approach for precipitation nowcasting», dans *Advances in neural information processing systems*, p. 802–810.

YEGAVIAN, R., B. LECLAIRE, F. CHAMPAGNAT, C. ILLOUL et G. LOSFELD. 2016, «Lucas–kanade fluid trajectories for time-resolved PIV», *Measurement science and Technology*, vol. 27, nº 8, p. 084 004.

YU, F., V. KOLTUN et T. A. FUNKHOUSER. 2017, «Dilated residual networks», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 3.

ZABIH, R. et J. WOODFILL. 1994, «Non-parametric local transforms for computing visual correspondence», dans *European conference on computer vision*, Springer, p. 151–158.

ZACH, C., T. POCK et H. BISCHOF. 2007, «A duality based approach for realtime TV-L1 optical flow», *Pattern Recognition*, p. 214–223.

ŽBONTAR, J. et Y. LECUN. 2016, «Stereo matching by training a convolutional neural network to compare image patches», *The journal of machine learning research*, vol. 17, nº 1, p. 2287–2318.

ZHAO, S., Y. SHENG, Y. DONG, E. I. CHANG, Y. XU et collab.. 2020, «MaskFlownet : Asymmetric feature matching with learnable occlusion mask», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 6278–6287.

Annexe A

Entraînement d'une méthode d'estimation de flot optique par apprentissage profond

Sommaire

Nous détaillons ici certains points — qui n'ont pas trouvé leur place dans le texte principal — de l'entraînement d'une méthode d'estimation de flot optique par apprentissage profond. Nous nous intéressons uniquement à un entraînement supervisé, *ie* n'utilisant que des données d'entraînement pour lesquelles on dispose de la VT du flot optique.

L'entraînement consiste à optimiser les paramètres d'un réseau de neurones convolutif, de sorte à minimiser une fonction de coût calculant l'erreur entre la prédiction réalisée par le réseau et la VT, pour les exemples d'entraînement qu'on aura choisis. On utilise pour cela un optimiseur, algorithme chargé de mettre à jour itérativement les paramètres du réseau à mesure qu'on parcourt les données d'entraînement. Cet algorithme optimise les paramètres du réseau, à chaque itération, en tenant compte d'un petit ensemble d'exemples d'apprentissage appelé *mini-batch*. Le nombre d'itérations par *epoch* (un *epoch* correspond à parcourir le jeu de données en entier) dépend donc du nombre d'exemples par *mini-batch*.

A.1 Réseaux de neurones utilisés pour le flot optique

Les architectures FlowNet et PWC-Net, qui ont servi de base à de nombreux autres travaux, sont construites comme un enchaînement de couches de convolution. Une opération non linéaire "Leaky ReLU" est placée après chaque convolution (sauf s'il s'agit de la dernière convolution d'un bloc destiné à estimer le flot optique). Ces architectures n'utilisent pas de *batch norm*, elles n'utilisent pas non plus de *drop out*.

A.2 Optimiseur et fonction de coût

Dès l'article de DOSOVITSKIY et collab. [2015], les méthodes d'estimation de flot par apprentissage profond sont entraînées en utilisant l'optimiseur Adam (KINGMA et BA [2014]), avec pour paramètres $\beta_1 = 0.9$ et $\beta_2 = 0.999$.

Comme nous l'avons vu au chapitre 2, la fonction de coût utilisée pour entraîner les paramètres Θ du réseau est la somme sur les L échelles de l'EPE. Plus précisément, ces méthodes ajoutent à la fonction de coût un terme dit de "weight decay", permettant de limiter le sur-apprentissage (la sur-spécialisation sur l'ensemble d'entraînement) en introduisant une régularisation sur les poids du réseau.

$$\mathscr{L}(\Theta) = \sum_{l=l_0}^{L} \alpha_l \sum_{\boldsymbol{x}} \left\| \boldsymbol{u}_{\Theta}^{(l)}(\boldsymbol{x}) - \boldsymbol{u}_{\mathrm{VT}}^{(l)}(\boldsymbol{x}) \right\| + \gamma \|\Theta\|^2 \tag{A.1}$$

où $\|\cdot\|$ représente la norme euclidienne, $\boldsymbol{u}_{\Theta}^{(l)}(\boldsymbol{x})$ le flot estimé à l'échelle l et $\boldsymbol{u}_{\mathrm{VT}}^{(l)}(\boldsymbol{x})$ la vérité terrain du flot optique, sous-échantillonnée à la taille image correspondant à l'échelle l, au pixel de coordonnées $\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix}$. La pondération associée au *weight decay* est $\gamma = 0.0004$, les pondérations associées à chacune des échelles sont $\alpha_6 = 0.32$, $\alpha_5 = 0.08$, $\alpha_4 = 0.02$, $\alpha_3 = 0.01$ et $\alpha_2 = 0.005$.

A.3 Entraînement

A.3.1 Jeux de données synthétiques avec vérité terrain du flot optique

FlyingChairs

Proposé par DOSOVITSKIY et collab. [2015], FlyingChairs comporte 22 232 paires d'images pour l'entraînement, et 1280 pour l'évaluation. Les mouvements sont des transformations affines 2D des chaises et du fond. Avec des *batchs* de taille 8, les 22 232 exemples d'entraînement correspondent à 2779 itérations par *epoch*.

FlyingThings3D *subset*

Le jeu de données FlyingThings3D, proposé par MAYER et collab. [2016], comporte des séquences générées avec Blender. Il s'agit de modèles 3D d'objets variés et qui se déplacement dans un environnement 3D. Les mouvements sont parfois assez violents, d'amplitude nettement plus importante que dans FlyingChairs.

Le *subset* est un sous-ensemble de 2183 séquences de 9 ou 10 images (21818 images au total) pour l'entraînement et 425 séquences pour la validation.

Pour un entraînement biframe, toutes les paires consécutives sont considérées. Cela représente 19636 exemples d'apprentissage, ce qui correspond, avec des *batch* de taille 4, à 4909 itérations par *epoch*.

A.3.2 Normalisation

Avant d'être présentées au réseau, les images sont normalisées pour prendre des valeurs comprises entre 0 et 1. En général, ceci est fait simplement en divisant toutes les valeurs de l'image par 255, pour des images codées sur 8 bits.

Il est courant de diviser la vérité terrain du flot optique par 20 avant de l'utiliser pour superviser l'entraînement du réseau. Ainsi, lorsque le réseau est utilisé en pratique, sa sortie doit être multipliée par 20 pour obtenir le bon résultat. Ceci est peu évoqué dans la littérature (SUN et collab. [2018a] le mentionnent mais sans plus de détails) mais apparaît dans plusieurs implémentations *open source* comme celles de SUN et collab. [2018a] ou de HUR et ROTH [2019]. Les images d'entrées étant normalisées entre 0 et 1, la raison de cette division du flot VT par 20 est probablement d'obtenir des valeurs de sortie du réseau du même ordre de grandeur. La valeur 20 est sans doute un peu arbitraire mais semble correspondre approximativement à la valeur moyenne des amplitudes des déplacements sur MPI Sintel.

A.3.3 Augmentations de données

Des transformations aléatoires sont appliquées aux données d'entraînement afin d'en augmenter artificiellement la diversité. Ainsi lorsqu'on parcourt plusieurs fois le même jeu de données, on ne verra pas strictement les mêmes exemples, grâce à ces transformations aléatoires.

Deux types d'augmentation de données sont utilisés pour le flot : les géométriques et les photométriques. Les géométriques sont des transformations affines, symétries horizontales et/ou verticales, et des *crops* (une sous-zone de l'image est sélectionnée aléatoirement). Les photométriques sont des changements de l'intensité, du contraste, du gamma et des ajouts de bruit.

Pour les transformations géométriques affines on peut appliquer des transformations différentes pour les deux images, et modifier les valeurs du flot VT en conséquence, cela permet d'augmenter également la variabilité des flots.

A.3.4 Stratégie d'entraînement

On entend par là le choix des données d'entraînement, l'ordre dans lequel on les présente, et le choix des hyper-paramètres.

Les travaux de ILG et collab. [2017] ont établi la stratégie d'entraînement reprise par les travaux plus récents (HUR et ROTH [2019]; NEORAL et collab. [2018]; SUN et collab. [2018b]). Le réseau est pré-entraîné sur FlyingChairs, puis entraîné sur FlyingThings3D *subset*, puis on réalise un *finetuning* soit sur Sintel soit sur KITTI, ou éventuellement sur les deux, selon le contexte applicatif visé.

Pré-entrainement sur FlyingChairs

Selon les travaux, on rencontre un pré-entraînement court S_{short}, ou plus long S_{long}. Dans les deux cas on utilise des *mini-batch* de taille 8 et un taux d'apprentissage initial de 10^{-4}.

Le scénario court S_{short} consiste à réaliser au total 600 000 itérations. Les 300 000 premières sont faites avec un taux d'apprentissage de 10^{-4}, puis celui-ci est divisé par 2 toutes les 100 000 itérations. Il s'agit de l'entraînement proposé par DOSOVITSKIY et collab. [2015] pour FlowNetS.

Le scénario S_{long} consiste à réaliser au total 1 200 000 itérations, les 400 000 premières avec un taux d'apprentissage de 10^{-4}, puis en le divisant par 2 toutes les 200 000 itérations. ILG et collab. [2017] ont proposé cet entraînement et ont montré qu'il donnait de meilleurs résultats que S_{short} à l'issu de la phase de pré-entraînement.

Entraînement sur FlyingThings3D *subset*

La deuxième phase consiste à entraîner le modèle, pré-entraîné sur FlyingChairs, sur le *subset* de FlyingThings3D. Pour cela on commence avec un taux d'apprentissage de 10^{-5}, pendant 200 000 itérations, puis on le divise par 2 toutes les 100 000 itérations. Au total cette phase dure 500 000 itérations et on utilise des *mini-batchs* de taille 4.

Finetuning sur Sintel/KITTI

On réalise enfin un *finetuning* sur Sintel ou KITTI. Pour Sintel on entraîne sur un mélange de Sintel Clean et de Sintel Final. Pour KITTI, il s'agit d'un mélange des versions 2012 et 2015. On utilise des *batchs* de taille 4. Pour cette phase, SUN et collab. [2018a] ont montré qu'on obtenait de meilleurs résultats en réaugmentant le taux d'apprentissage à mi-entraînement. Le scénario proposé pour le taux d'apprentissage est celui représenté sur la figure A.1.

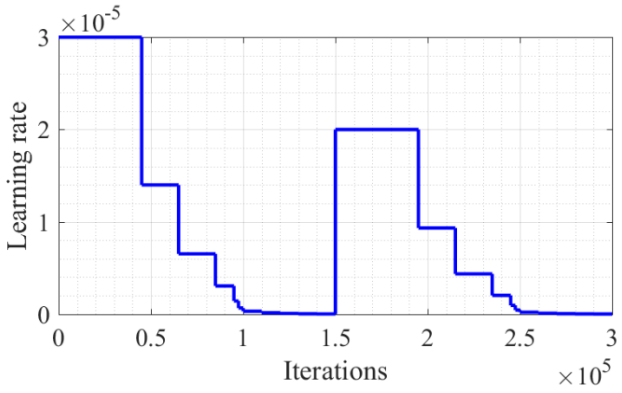

FIGURE A.1 – Évolution du taux d'apprentissage pendant le *finetuning*. Source : SUN et collab. [2018a].

A.4 Bruit d'entraînement

Afin de caractériser la répétabilité du processus d'entraînement, nous avons effectué 10 fois le même entraînement (FlyingChairs → FlyingThings3D *subset*) de l'architecture PWC-Net [1] et comparé les erreurs de validation obtenue sur l'ensemble d'entraînement de Sintel Final. La figure A.2 montre l'évolution de cette erreur de validation au cours de l'entraînement pour chacune des 10 réalisations. À la fin de l'entraînement, l'erreur moyenne sur toutes les réalisations est de 5.131 pixels, et l'écart-type est de 0.161 px. Cet écart-type caractérise un bruit d'entraînement, lors de la comparaison de méthodes d'estimation de flot optique par apprentissage profond, il faut donc considérer qu'un gain de l'ordre de ce bruit d'entraînement n'est pas significatif.

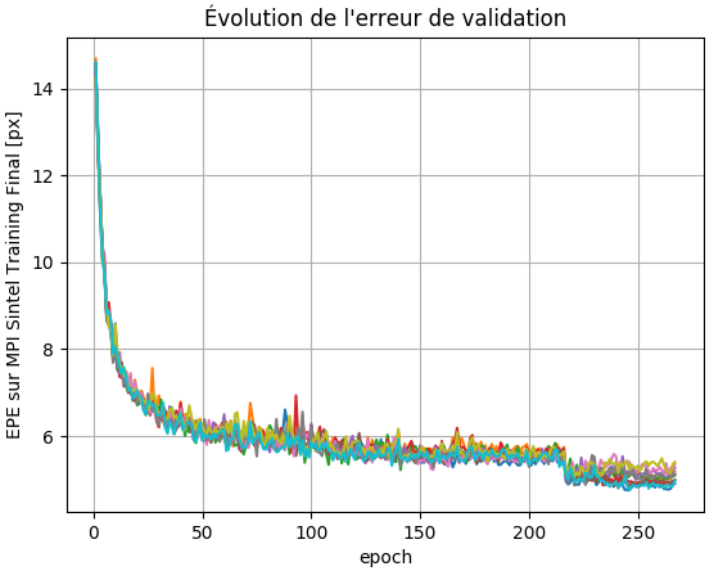

FIGURE A.2 – Pour 10 réalisations de l'entraînement de PWC-Net, évolution de l'erreur de validation (EPE) sur l'ensemble d'entraînement de Sintel Final, au cours de l'entraînement sur FlyingChairs → FlyingThings3D *subset*. Le changement de *dataset* d'entraînement a lieu à l'*epoch* 216.

1. Nous utilisons l'implémentation de https://github.com/visinf/irr.

A.5 Entraînement de STaRFlow

Nous donnons ici quelques détails supplémentaires sur l'entraînement de la méthode STaRFlow présentée dans le chapitre 5. La procédure d'entraînement pour cette méthode multiframe est expliquée en section 5.4. Nous donnons ici les courbes d'évolution de la fonction de coût et de l'erreur de validation pendant chaque phase de l'entraînement, et détaillons les temps d'entraînement, sur un GPU Nvidia V100. La **durée totale de l'entraînement** est de **439 heures**, soit **18 jours et 7 heures**.

A.5.1 Pré-entrainement sur FlyingChairs

Les courbes sont données en figure A.3. Cette phase a duré 135 heures. Pour cette phase chaque *epoch* correspond à 2779 itérations (pour des *batchs* de 8 exemples) et dure environ 35 minutes.

A.5.2 Entraînement sur FlyingThings3D *subset*

Les courbes correspondant à cette phase sont sur la figure A.4. Cette phase a duré 7 jours (168h), elle était censée durer plus longtemps mais s'est arrêtée prématurément pour des raisons techniques et nous ne l'avons réalisé que plus tard. Pour respecter la procédure annoncée en 5.4, il aurait fallu faire 110 epochs, soit 28 de plus. Pour cette phase chaque *epoch* correspond à 3818 itérations (pour des *batchs* de 4 exemples) et dure environ 2 heures.

On peut remarquer sur les courbes que les fonctions d'erreur n'ont pas encore atteint un plateau. En entraînant plus longtemps on pourrait sans doute continuer à gagner sur l'erreur de validation, et peut-être aussi sur les performances générales du réseau. Pour s'en assurer, on pourrait poursuivre l'entraînement en suivant l'évolution de l'erreur sur Sintel (par exemple) également.

A.5.3 *Finetuning* sur Sintel

La courbe d'évolution de la fonction de coût correspondant au *finetuning* sur Sintel est présentée en figure A.5. Comme nous entraînons sur la totalité des séquences, nous ne disposons pas d'ensemble de validation et ne présentons donc que l'évolution de la fonction de coût. Cet entraînement se passe en deux phases : la première sur la réunion de Sintel Clean et Sintel Final, la seconde sur Sintel Final uniquement. L'ensemble a pris 105 heures. Pour la première phase chaque *epoch* correspond à 498 itérations (pour des *batchs* de 4 exemples) et dure environ 16 minutes. Pour la seconde, chaque *epoch* correspond à 249 itérations (toujours avec 4 images par *batch*) et dure environ 8 minutes.

La valeur de la fonction de coût est plus élevée pendant la seconde phase, deux raisons expliquent ce phénomène :

— Le taux d'apprentissage suit l'évolution présentée en figure A.1, il réaugmente donc au début de cette seconde phase. Ce pas de descente de gradient brusquement élevé peut réaugmenter la valeur de la fonction de coût.

— Pendant la première phase en entraîne sur la réunion de Sintel Clean et Sintel Training et pendant la seconde sur Sintel Final uniquement. Sintel Final étant plus difficile, l'erreur y est plus élevée.

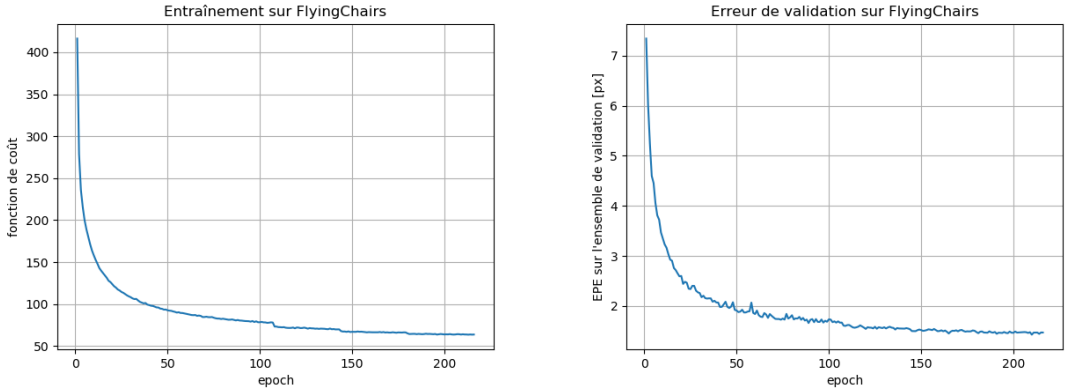

FIGURE A.3 – Évolution de la fonction de coût et de l'erreur de validation au cours de l'entraînement sur FlyingChairs.

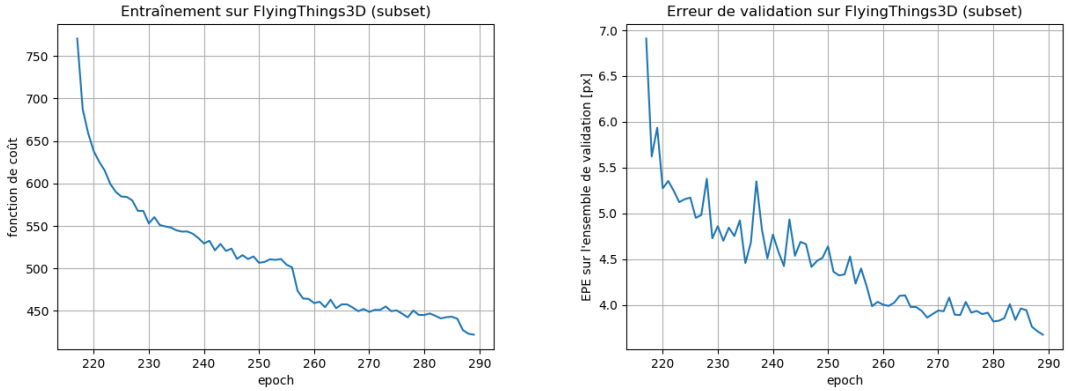

FIGURE A.4 – Évolution de la fonction de coût et de l'erreur de validation au cours de l'entraînement sur FlyingThings3D (*subset*).

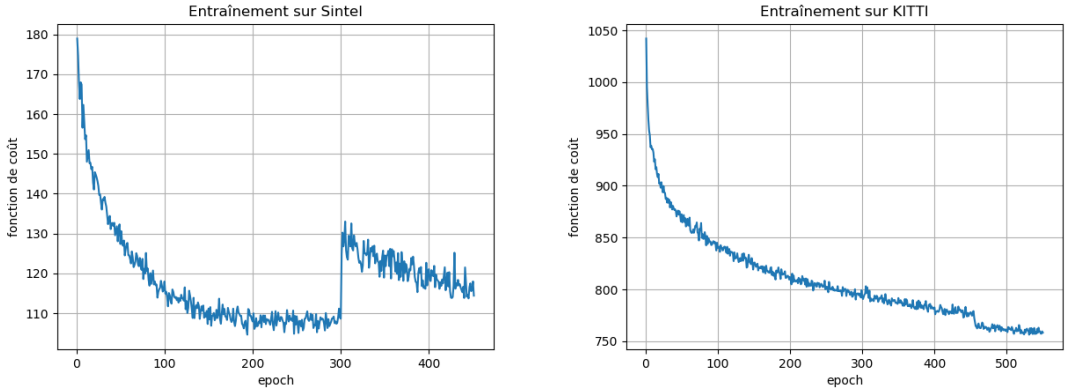

FIGURE A.5 – Évolution de la fonction de coût au cours du *finetuning* sur Sintel (à gauche) ou KITTI (à droite).

A.5.4 *Finetuning* sur KITTI

La courbe d'évolution de la fonction de coût correspondant au *finetuning* sur KITTI est présentée en figure A.5. Comme nous entraînons sur la totalité des séquences, nous ne disposons pas d'ensemble de validation et ne présentons donc que l'évolution de la fonction de coût. Cette phase a duré 31 heures. Pour cette phase chaque *epoch* correspond à 100 itérations (pour des *batchs* de 4 exemples) et dure un peu plus de 3 minutes.

A.5.5 Perspective : Réduire le temps d'entraînement

La durée totale d'entraînement de STaRFlow (mais aussi de ce type de méthode en général) est très longue. Nous n'avons pas cherché à optimiser la procédure d'entraînement de ce point de vue là, et sommes repartis du protocole indiqué dans la littérature récente du domaine. En analysant les courbes d'erreur au cours de l'entraînement, on aurait envie d'arrêter le pré-entraînement sur FlyingChairs (figure A.3) plus tôt.

A.6 Références

DOSOVITSKIY, A., P. FISCHER, E. ILG, P. HAUSSER, C. HAZIRBAS, V. GOLKOV, P. VAN DER SMAGT, D. CREMERS et T. BROX. 2015, «FlowNet : Learning optical flow with convolutional networks», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 2758–2766. II, III, IV

HUR, J. et S. ROTH. 2019, «Iterative residual refinement for joint optical flow and occlusion estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5754–5763. III, IV

ILG, E., N. MAYER, T. SAIKIA, M. KEUPER, A. DOSOVITSKIY et T. BROX. 2017, «FlowNet 2.0 : Evolution of optical flow estimation with deep networks», dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 6. IV

KINGMA, D. P. et J. BA. 2014, «Adam : A method for stochastic optimization», *arXiv preprint arXiv :1412.6980*. II

MAYER, N., E. ILG, P. HAUSSER, P. FISCHER, D. CREMERS, A. DOSOVITSKIY et T. BROX. 2016, «A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4040–4048. III

NEORAL, M., J. ŠOCHMAN et J. MATAS. 2018, «Continual occlusion and optical flow estimation», dans *Asian Conference on Computer Vision*, Springer, p. 159–174. IV

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018a, «Models matter, so does training : An empirical study of CNNs for optical flow estimation», *arXiv preprint arXiv :1809.05571*. III, IV, V

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018b, «PWC-Net : CNNs for optical flow using pyramid, warping, and cost volume», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 8934–8943. IV

Annexe B

Inversibilité de la matrice A de FOLKI-MF

Sommaire

Cette annexe détaille, et démontre, les conditions d'inversibilité de la matrice $\mathrm{A}_{\boldsymbol{k}}$ introduite dans le chapitre 3. Ces conditions d'inversibilité conditionnent les choix de modèles temporels possibles pour l'algorithme FOLKI-MF, présenté en section 3.3. Les notations utilisées ici sont celles du chapitre 3.
L'expression factorisée de la matrice $\mathrm{A}_{\boldsymbol{k}}$

$$\mathrm{A}_{\boldsymbol{k}} = \sum_{n=1}^{N} \mathrm{T}_n^{\mathsf{T}} \left(\sum_{\boldsymbol{k}' \in \mathscr{W}(\boldsymbol{k})} \nabla I_{ref}(\boldsymbol{k}') \nabla I_{ref}(\boldsymbol{k}')^{\mathsf{T}} \right) \mathrm{T}_n \tag{3.19}$$

fait intervenir la matrice, de dimension 2×2, de covariance des gradients, qu'on notera :

$$\begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix} = \left(\sum_{\boldsymbol{k}' \in F(\boldsymbol{k})} \nabla I_{ref}(\boldsymbol{k}') (\nabla I_{ref})^{\mathsf{T}}(\boldsymbol{k}') \right) \tag{B.1}$$

On peut donc réécrire cette matrice comme

$$\mathrm{A}_{\boldsymbol{k}} = \sum_{n=1}^{N} \begin{pmatrix} \boldsymbol{E}_n^{\mathsf{T}} & 0 \\ 0 & \boldsymbol{H}_n^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix} \begin{pmatrix} \boldsymbol{E}_n^{\mathsf{T}} & 0 \\ 0 & \boldsymbol{H}_n^{\mathsf{T}} \end{pmatrix} \tag{B.2}$$

et montrer qu'elle est symétrique ($\mathrm{A}_{\boldsymbol{k}} = \mathrm{A}_{\boldsymbol{k}}^{\mathsf{T}}$), et semi-définie positive c'est-à-dire que quelques soient $\boldsymbol{x}$ et $\boldsymbol{y}$, vecteurs de $\mathbb{R}^d$, dont au moins l'un des deux est non nul on a :

$$\begin{pmatrix} \boldsymbol{x}^{\mathsf{T}} & \boldsymbol{y}^{\mathsf{T}} \end{pmatrix} \mathrm{A}_{\boldsymbol{k}} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \sum_{n=1}^{N} \begin{pmatrix} \boldsymbol{x}^{\mathsf{T}} \boldsymbol{E}_n & \boldsymbol{y}^{\mathsf{T}} \boldsymbol{H}_n \end{pmatrix} \begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix} \begin{pmatrix} \boldsymbol{E}_n^{\mathsf{T}} \boldsymbol{x} \\ \boldsymbol{H}_n^{\mathsf{T}} \boldsymbol{y} \end{pmatrix} \geq 0 \tag{B.3}$$

car quelque soit le vecteur $\boldsymbol{u}$ de $\mathbb{R}^2$

$$\boldsymbol{u}^{\mathsf{T}} \begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix} \boldsymbol{u} = \sum_{\boldsymbol{k}' \in F(\boldsymbol{k})} \left\| \nabla I_{ref}^{\mathsf{T}}(\boldsymbol{k}') \boldsymbol{u} \right\|_2^2 \geq 0 \tag{B.4}$$

La question de l'inversibilité de $\mathrm{A}_{\boldsymbol{k}}$ se ramène donc à celle de sa définie positivité.

B.1 Cas où la matrice de covariance des gradients est inversible

Étudions le cas où la matrice de covariance des gradients est inversible (et par conséquent définie positive). Ce cas correspond à $\alpha\gamma \neq \beta^2$.

Theorème 1. *Dans ce cas, la matrice* $\mathrm{A}_{\boldsymbol{k}}$ *est définie positive (et donc inversible) si et seulement si les jeux de vecteurs* $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ *et* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d$ *définissent chacun une base d'un sous-espace de dimension* d *dans* $\mathbb{R}^N$, ie *les matrices*

$$\mathrm{E} = \begin{pmatrix} \boldsymbol{E}_1^{\mathsf{T}} \\ \vdots \\ \boldsymbol{E}_N^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{e}_1[1] & \cdots & \boldsymbol{e}_d[1] \\ \vdots & \ddots & \vdots \\ \boldsymbol{e}_1[N] & \cdots & \boldsymbol{e}_d[N] \end{pmatrix} \text{ et } \mathrm{H} = \begin{pmatrix} \boldsymbol{H}_1^{\mathsf{T}} \\ \vdots \\ \boldsymbol{H}_N^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{h}_1[1] & \cdots & \boldsymbol{h}_d[1] \\ \vdots & \ddots & \vdots \\ \boldsymbol{h}_1[N] & \cdots & \boldsymbol{h}_d[N] \end{pmatrix}, \tag{B.5}$$

de dimension $N \times d$, *sont de rang* d.

Démonstration. $\mathrm{A}_{\boldsymbol{k}}$ est définie positive si et seulement si quelques soient $\boldsymbol{x}$ et $\boldsymbol{y}$, vecteurs de $\mathbb{R}^d$, dont au moins l'un des deux est non nul on a :

$$\begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \mathrm{A}_{\boldsymbol{k}} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \sum_{n=1}^{N} \begin{pmatrix} \boldsymbol{x}^\top \boldsymbol{E}_n & \boldsymbol{y}^\top \boldsymbol{H}_n \end{pmatrix} \begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix} \begin{pmatrix} \boldsymbol{E}_n^\top \boldsymbol{x} \\ \boldsymbol{H}_n^\top \boldsymbol{y} \end{pmatrix} > 0 \tag{B.6}$$

Ce qui équivaut, comme $\begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix}$ est définie positive, à dire qu'il existe au moins un n tel que

$$\begin{pmatrix} \boldsymbol{E}_n^\top \boldsymbol{x} \\ \boldsymbol{H}_n^\top \boldsymbol{y} \end{pmatrix} \neq 0 \tag{B.7}$$

autrement dit

$$\mathrm{A}_{\boldsymbol{k}} \text{ est définie positive} \iff \forall \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \neq 0,\ \begin{pmatrix} \mathrm{E}\boldsymbol{x} \\ \mathrm{H}\boldsymbol{y} \end{pmatrix} \neq 0 \tag{B.8}$$

$$\iff \{\forall \boldsymbol{x} \neq 0,\ \mathrm{E}\boldsymbol{x} \neq 0\} \text{ et } \{\forall \boldsymbol{y} \neq 0,\ \mathrm{H}\boldsymbol{y} \neq 0\} \tag{B.9}$$

Développons $\{\forall \boldsymbol{x} \neq 0,\ \mathrm{E}\boldsymbol{x} \neq 0\}$, l'autre est analogue

$$\forall \boldsymbol{x} \neq 0,\ \mathrm{E}\boldsymbol{x} \neq 0 \iff \forall \boldsymbol{x} \neq 0,\ \boldsymbol{x}^\top \mathrm{E}^\top \mathrm{E}\boldsymbol{x} = \|\mathrm{E}\boldsymbol{x}\|^2 \neq 0 \tag{B.10}$$

$$\iff \mathrm{E}^\top \mathrm{E} \text{ est définie positive} \tag{B.11}$$

$$\iff \mathrm{E}^\top \mathrm{E} \text{ est inversible} \tag{B.12}$$

$$\iff \mathrm{rg}(\mathrm{E}) = d \tag{B.13}$$

□

B.2 Cas où la matrice de covariance des gradients n'est pas inversible

Ce cas correspond à $\alpha\gamma = \beta^2$, la matrice de covariance des gradients est de rang au plus 1. Le cas où elle est de rang 0 ($\alpha = \beta = \gamma = 0$) est désespéré : aucune chance dans ce cas d'inverser $\mathrm{A}_{\boldsymbol{k}}$. On supposera donc ici $\begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix}$ de rang exactement égal à 1, ce qui suppose que $\alpha > 0$ ou $\gamma > 0$.

Dans ce cas on peut réécrire la matrice

$$\begin{pmatrix} \alpha_{\boldsymbol{k}} & \beta_{\boldsymbol{k}} \\ \beta_{\boldsymbol{k}} & \gamma_{\boldsymbol{k}} \end{pmatrix} = \begin{pmatrix} \sqrt{\alpha_{\boldsymbol{k}}} \\ \sqrt{\gamma_{\boldsymbol{k}}} \end{pmatrix} \begin{pmatrix} \sqrt{\alpha_{\boldsymbol{k}}} \\ \sqrt{\gamma_{\boldsymbol{k}}} \end{pmatrix}^\top \tag{B.14}$$

Et on a donc (réécriture de (B.3) dans ce cas)

$$\begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \mathrm{A}_{\boldsymbol{k}} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \sum_{n=1}^{N} \left\| \begin{pmatrix} \sqrt{\alpha_{\boldsymbol{k}}} \\ \sqrt{\gamma_{\boldsymbol{k}}} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{E}_n^\top \boldsymbol{x} \\ \boldsymbol{H}_n^\top \boldsymbol{y} \end{pmatrix} \right\|_2^2 \tag{B.15}$$

Cette expression est nulle si et seulement si $\sqrt{\alpha_{\boldsymbol{k}}}(\mathrm{E}\boldsymbol{x}) + \sqrt{\gamma_{\boldsymbol{k}}}(\mathrm{H}\boldsymbol{y}) = 0$. On peut alors distinguer deux cas :

— α ou γ est nul, il s'agit du cas où la texture est orientée verticalement ou horizontalement. Supposons $\alpha \neq 0$ et $\gamma = 0$, on a alors

$$\sqrt{\alpha_{\boldsymbol{k}}}(\mathrm{E}\boldsymbol{x}) + \sqrt{\gamma_{\boldsymbol{k}}}(\mathrm{H}\boldsymbol{y}) = \sqrt{\alpha_{\boldsymbol{k}}}(\mathrm{E}\boldsymbol{x}) \tag{B.16}$$

qui peut être nul quelque soit $\boldsymbol{y}$. Dans ce cas la matrice $\mathrm{A}_{\boldsymbol{k}}$ n'est pas inversible (car (B.3) peut-être nulle sans que $\boldsymbol{x}$ et $\boldsymbol{y}$ soient tous deux nuls).

— ni α ni γ n'est nul, on peut donc poser $\boldsymbol{x}' = \sqrt{\alpha_{\boldsymbol{k}}}\boldsymbol{x}$ et $\boldsymbol{y}' = \sqrt{\gamma_{\boldsymbol{k}}}\boldsymbol{y}$ qui ne sont nuls que si $\boldsymbol{x}$ et $\boldsymbol{y}$ sont nuls.

$$\sqrt{\alpha_{\boldsymbol{k}}}(\mathrm{E}\boldsymbol{x}) + \sqrt{\gamma_{\boldsymbol{k}}}(\mathrm{H}\boldsymbol{y}) = \mathrm{E}\boldsymbol{x}' + \mathrm{H}\boldsymbol{y}' = \begin{pmatrix} \mathrm{E} & \mathrm{H} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \end{pmatrix} \tag{B.17}$$

Ainsi $\mathrm{A}_{\boldsymbol{k}}$ est inversible si et seulement si le noyau de $\begin{pmatrix} \mathrm{E} & \mathrm{H} \end{pmatrix}$ est réduit au vecteur nul, autrement dit si $\begin{pmatrix} \mathrm{E} & \mathrm{H} \end{pmatrix}$ est de rang $2d$ (avec $N \geq 2d$).

Donc pour avoir l'inversibilité de $\mathrm{A}_{\boldsymbol{k}}$ dans ce cas il faut

— Que la texture ne soit pas orientée selon l'un des axes horizontal ou vertical

— Avoir suffisamment d'instants ($N \geq 2d$)

— Que les bases horizontale (colonnes de E) et verticale (colonnes de H) soient complètement indépendantes.

Cette dernière condition paraît peu probable dans la pratique.

B.3 Bilan

— Si $\alpha\gamma \neq \beta^2$ alors $\mathrm{A}_{\boldsymbol{k}}$ est inversible dès lors que les jeux de vecteurs $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ et $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d$ définissent chacun une base d'un sous-espace de dimension d.

— Si $\alpha\gamma = \beta^2$ et $\alpha = 0$ et/ou $\gamma = 0$ alors $\mathrm{A}_{\boldsymbol{k}}$ n'est pas inversible.

— Si $\alpha\gamma = \beta^2$ et $\alpha \neq 0$ et $\gamma \neq 0$ alors $\mathrm{A}_{\boldsymbol{k}}$ n'est inversible que si l'ensemble des conditions suivantes sont réunies (peu probable en pratique) :

  — la texture n'est orientée ni selon l'axe horizontal ni selon l'axe vertical

  — $N \geq 2d$, *ie* si on a suffisamment d'instants par rapport à la dimension souhaitée pour les bases

  — les bases horizontale (colonnes de E) et verticale (colonnes de H) sont complètement indépendantes.

Rappelons que le cas $\alpha\gamma = \beta^2$ correspond au cas d'un manque de texture, qui pose problème de manière générale aux algorithmes locaux. Il n'est donc pas étonnant que, dans ce cas, la matrice $\mathrm{A}_{\boldsymbol{k}}$ ne soit pas inversible. En pratique, dans le contexte de la métrologie, on travaille avec des données bien texturées, on suppose donc que $\alpha\gamma \neq \beta^2$.

Annexe C

Recalage hétérogène d'images de télédétection par apprentissage profond

Sommaire

C.1 Introduction

Le mouvement *open source* récent en télédétection, notamment avec le programme Copernicus, conduit à la disponibilité croissante d'images issues de modalités d'acquisition différentes. Il devient possible d'imaginer des méthodes tirant profit de l'information complémentaire de ces données. Nous nous intéressons plus particulièrement à l'exploitation conjointe d'images optique et radar (SAR). Ces images, étant acquises par des capteurs différents et dans des conditions de prise de vue différentes, doivent d'abord être recalées. Une grande part de ce recalage est réglé par une opération de "georegistration" qui s'appuie sur les données associées aux images (position/orientation du capteur, datation, étalonnage) et un modèle de surface pour les projeter en géométrie terrain. Cependant, il reste des erreurs de localisation, notamment du fait de l'incertitude sur le relief ce qui justifie une phase de recalage non rigide, consistant en l'estimation d'un champ de déplacements dense.

L'aspect très différent des images issues des deux modalités, en particulier à cause de la présence d'inversions de contraste, rend la tâche très difficile pour les techniques usuelles de mise en correspondance. GeFolki (Brigot et collab. [2016]), une méthode développée à l'ONERA avant nos travaux, vise à réaliser le recalage d'images de télédétection, éventuellement issues de physiques d'acquisition différentes. Il s'agit d'une extension de l'algorithme FOLKI, appliquant un filtre de rang (Zabih et Woodfill [1994]) aux images d'entrée et prenant en compte l'existence d'inversions de contraste dans le calcul du critère de similarité. Cette méthode a donné de bons résultats dans le cas LIDAR/SAR, mais est mise en difficulté dans le cas optique/SAR.

Nous avons élaboré, avec Élise Koeniguer, Fabrice Janez et Benjamin Le Teurnier (qui était stagiaire dans notre équipe) une méthode basée sur l'apprentissage profond pour réaliser le recalage optique/SAR sur des données de télédétection. Pour cela nous avons constitué une base de données annotée, sur laquelle nous entraînons un réseau de neurones convolutif.

C.2 Données d'entraînement

Pour réaliser l'entraînement supervisé d'une méthode basée sur l'apprentissage profond nous avons avant tout besoin de données annotées pour la tâche souhaitée. Pour cela nous avons constitué une base de données optique-radar, en s'appuyant sur des données *open source* satellites et aéroportées. Chaque exemple d'apprentissage est composé d'une image optique et d'une image SAR superposables, ainsi que d'un modèle numérique de terrain à partir duquel un champ de déplacements est simulé et utilisé pour rééchantillonner une des deux images. La méthode prend en entrée ces deux images et doit produire en sortie un flot optique le plus proche possible du champ simulé selon une certaine fonction de coût. Les processus de génération des exemples d'apprentissage et de l'entraînement du réseau sont illustrés sur la figure C.1.

C.2.1 Origine des images

Pour constituer la base d'entraînement, nous avons collecté des données images issues de deux origines :

— des images satellitaires de Sentinel 1 (SAR) et Sentinel 2 (optique) du programme européen Copernicus.

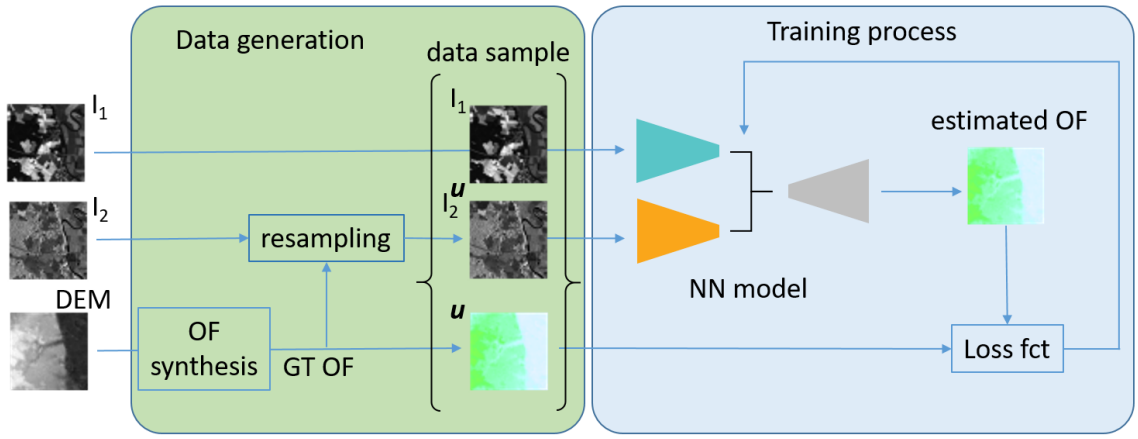

FIGURE C.1 – Génération des exemples d'apprentissage et entraînement du réseau de neurones. À gauche, on génère un champ de déplacement VT à partir du modèle numérique de terrain (DEM), puis à partir d'un couple optique/SAR superposable et de ce champ de déplacement on génère un exemple d'apprentissage en rééchantillonnant l'une des images. Le même champ de déplacement VT est utilisé pour pénaliser l'estimation du réseau au moyen d'une fonction de coût et, ainsi, superviser son entraînement.

— des images aériennes des programmes américains UAVSAR (Un-inhabited Aerial Vehicle Synthetic Aperture Radar) et NAIP (National Agriculture Imagery Program).

Ces données diffèrent par leur échantillonnage spatial, qui est de 10 mètres pour les images satellitaires et de 5 mètres pour les images aériennes; par les paysages qu'elles contiennent, en France pour les images satellitaires et aux États-Unis pour les images aériennes; et par la bande de longueur d'onde utilisée pour l'image SAR, bande C pour Sentinel 1 et bande L pour UAVSAR. Les emprises et un exemple d'images sont montrés sur les figures C.2 et C.3.

Les bases satellitaire et aérienne sont, chacune, séparées en un jeu d'entraînement et un jeu de validation. La base satellitaire est composée de 525 images 2000×2000 [1] pour l'entraînement et de 16 images 448×448 de validation. Dans la base aérienne on dispose de 1220 images 1024×1024 pour l'entraînement et 203 pour la validation.

C.2.2 Génération des champs de déplacements

Afin de pouvoir superviser l'entraînement, nous générons pour chaque exemple une vérité terrain du champ de déplacements. Nous proposons une méthode pour imiter simplement le type de champ de déplacements rencontré en pratique dans le contexte du recalage d'images de télédétection, partant du principe que les erreurs de recalage sont essentiellement liées au relief, en utilisant le modèle numérique de terrain (DEM pour "Digital Elevation Model"). Ci-après, nous expliquons séparément comment nous générons la norme et la direction du déplacement en chaque pixel.

Génération de la norme du déplacement

Pour chaque exemple (regroupant pour la même zone l'image optique, l'image SAR et le DEM) on tire aléatoirement selon une distribution uniforme :

1. En pratique on montre au réseau de neurones des images plus petites (448×448) en prenant des *crops* dans ces images.

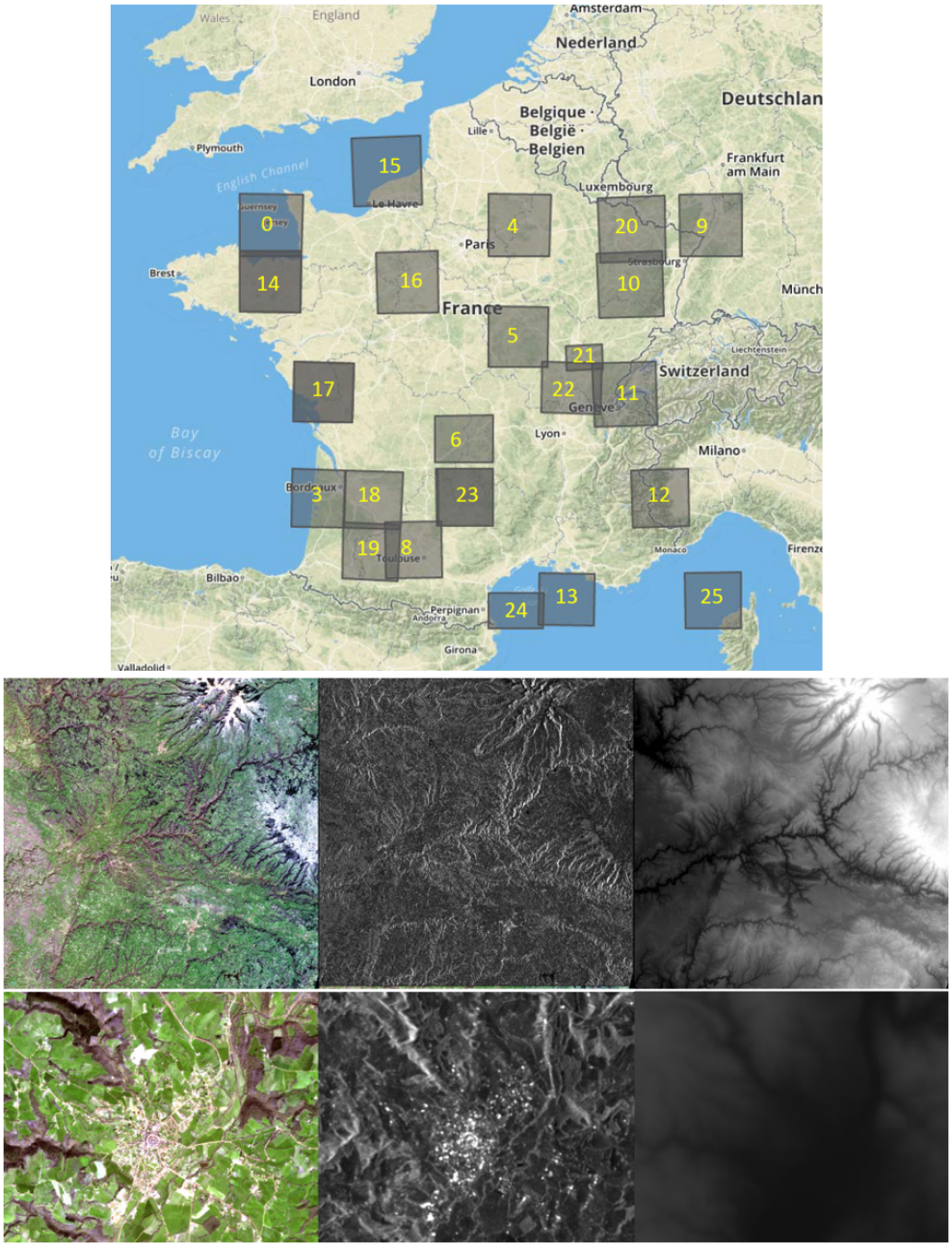

FIGURE C.2 – Base d'images satellitaires. En haut : emprises sur le territoire français pour la base satellitaire. En bas : exemple d'image optique, image SAR et modèle numérique de terrain (et versions zoomées des mêmes images en dessous).

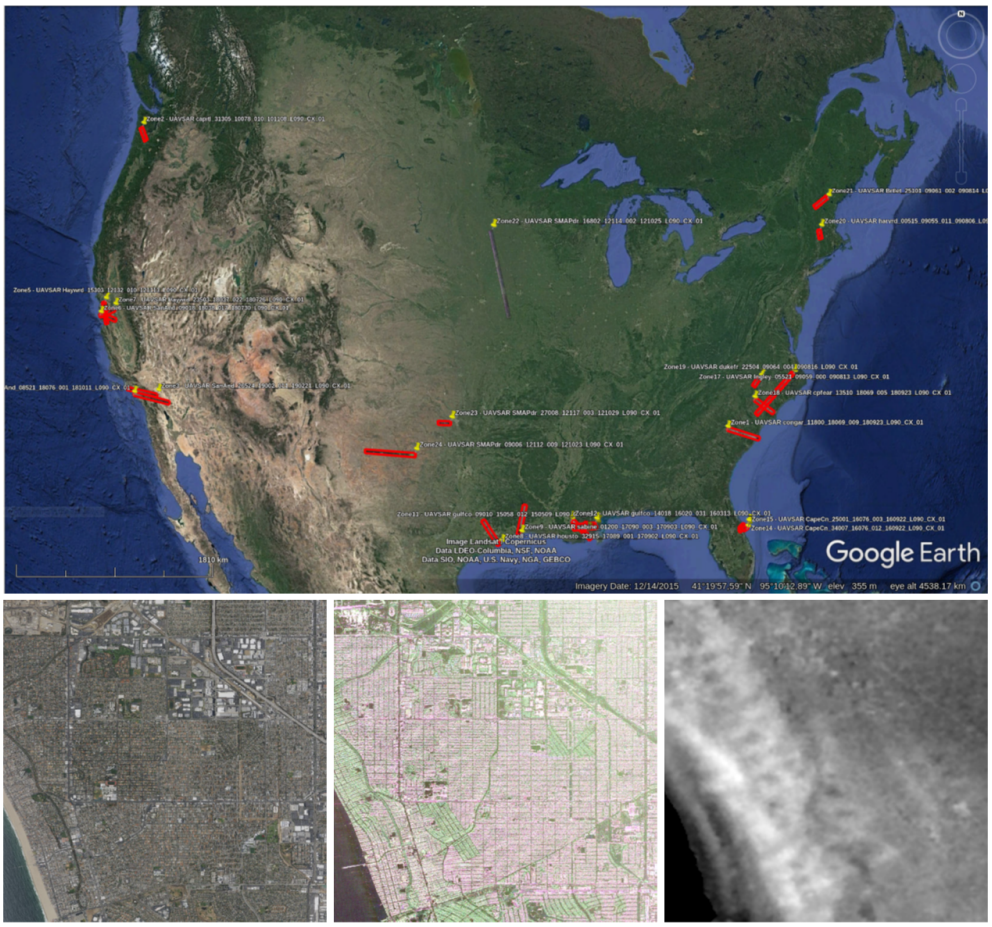

FIGURE C.3 – Base d'images aériennes. En haut : emprises sur le territoire américain pour la base aérienne. En bas : exemple d'image optique, image SAR et modèle numérique de terrain.

— une amplitude de déplacement minimum a_{min} entre 0 et 3 pixels.

— une amplitude de déplacement maximum a_{max} entre 3 et 10 pixels.

La norme du déplacement $\boldsymbol{u}$ évolue entre ces deux valeurs, en suivant les variations spatiales du DEM. La norme au pixel de coordonnées $\boldsymbol{x}$ est donnée par :

$$\|\boldsymbol{u}(\boldsymbol{x})\| = a_{min} + \frac{\mathrm{DEM}(\boldsymbol{x}) - \mathrm{DEM}_{min}}{\mathrm{DEM}_{max} - \mathrm{DEM}_{min}} \times (a_{max} - a_{min}) \tag{C.1}$$

Génération de la direction du déplacement

Pour chaque exemple on tire aléatoirement selon une distribution uniforme :

— un angle α entre 0 et 2π.

— une variation d'angle maximum δ_{max} entre $-\pi$ et π.

La direction du déplacement au pixel de coordonnées $\boldsymbol{x}$ est donnée par :

$$\arg(\boldsymbol{u}(\boldsymbol{x})) = \alpha + \frac{\mathrm{DEM}(\boldsymbol{x}) - \mathrm{DEM}_{min}}{\mathrm{DEM}_{max} - \mathrm{DEM}_{min}} \times \delta_{max} \tag{C.2}$$

C.3 Architecture du réseau de neurones

Nous avons proposé une modification de l'architecture PWC-Net (SUN et collab. [2018]), afin de l'adapter au cas d'images issues de modalités différentes. PWC-Net, dont une représentation simplifiée est présentée sur la figure C.4, utilise un même encodeur pour les deux images d'entrée (il s'agit d'un réseau siamois). Notre variante, appelée PWC-Net-multimodal, utilise deux encodeurs, ayant une structure identique à celle de l'encodeur de PWC-Net, mais pouvant apprendre des poids distincts, afin qu'ils puissent effectuer des opérations différentes pour l'image optique et l'image SAR. En sortie, comme dans PWC-Net, un décodeur commun estime le flot optique entre les deux images.

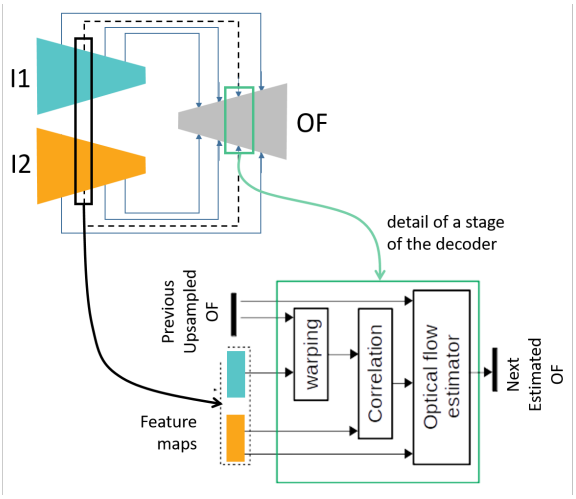

FIGURE C.4 – Représentation simplifiée de l'architecture PWC-Net. L'algorithme initialement proposé par SUN et collab. [2018] partage les poids des deux encodeurs. Nous proposons une version multimodale dans laquelle deux encodeurs, avec des poids différents, sont entraînés (l'un dédié aux images optiques, l'autre aux images SAR).

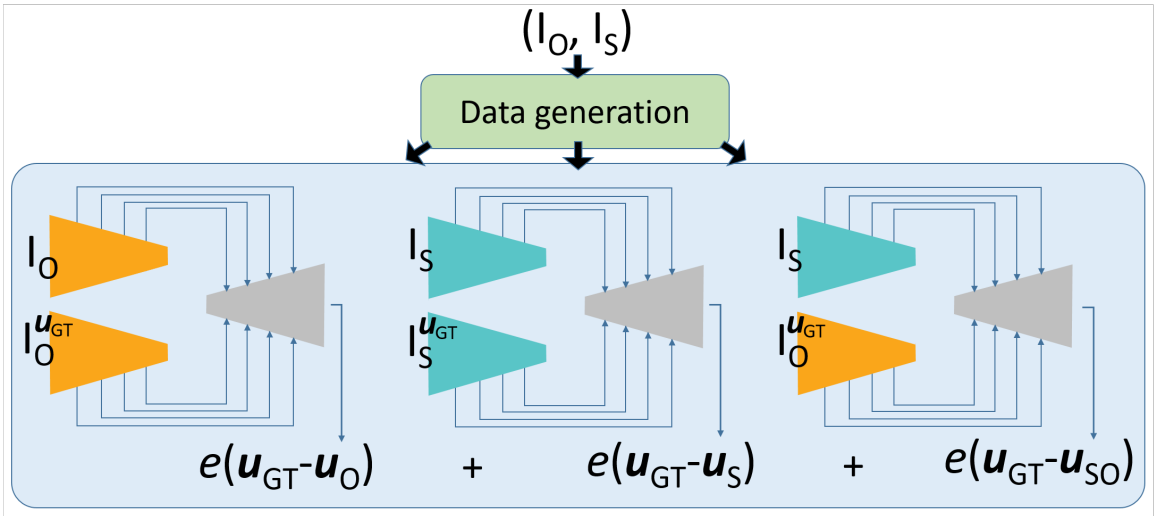

FIGURE C.5 – Stratégie d'entraînement à 3 tâches : recalage optique/optique, SAR/SAR et optique/SAR. Chacune des trois estimations, correspondant à chacune de ces paires d'images, donne lieu à un terme dans la fonction de coût comparant cette estimation à la VT (ou GT pour "Ground Truth").

C.4 Fonction de coût pour l'entraînement

Le réseau est entraîné en utilisant la même fonction de coût que celle utilisée pour entraîner PWC-Net, basée sur l'erreur *endpoint*, et rappelée dans l'annexe A. Cependant, entraîner le réseau directement pour la tâche de recalage hétérogène ne fonctionne pas. Pour guider l'apprentissage nous avons proposé d'entraîner le réseau simultanément pour les tâches de recalages optique/optique, SAR/SAR et optique/SAR. En partant d'une paire d'images optique/SAR superposable, on rééchantillonne les deux images selon le champ de déplacements simulé afin d'obtenir une paire d'images optiques et une paire d'images SAR correspondant à la même zone et au même mouvement. Comme illustré sur la figure C.5, on estime, avec PWC-Net-multimodal, 3 flots optiques (correspondant aux paires optique/optique, SAR/SAR et optique/SAR) et on calcule l'erreur par rapport au champ de déplacements simulé de chacun des trois flots. On optimise le réseau en cherchant à minimiser la somme des trois fonctions de coûts.

C.5 Expériences et résultats

Nous présentons et comparons ici les résultats obtenus avec GeFolki et avec PWC-Net-multimodal, sur les ensembles de validation des bases d'images satellitaires et aériennes. Tous les résultats donnés pour PWC-Net-multimodal sont obtenus avec le même modèle issu d'un apprentissage sur un mélange des ensembles d'entraînements des bases satellitaire et aérienne.

C.5.1 Choix des paramètres de GeFolki

GeFolki nécessite un réglage supplémentaire par rapport à FOLKI, qui est le rayon de la fenêtre utilisé pour le filtre de rang. Celui-ci est réglé à 4 pixels dans nos expériences. Une autre différence est qu'il est possible, avec GeFolki, de réaliser les premières itéra-

TABLEAU C.1 – Erreurs *endpoint* (en pixels) moyennes sur les ensembles de validation des bases d'images satellitaires et aériennes, pour GeFolki et PWC-Net-multimodal.

Méthode	satellitaire	aérienne
GeFolki optique/optique	0.19	0.18
GeFolki optique/SAR	2.11	2.16
PWC-Net-multimodal optique/SAR	0.72	1.08

tions avec une grande taille de fenêtre, puis de diminuer cette taille de fenêtre pour les itérations suivantes. Nous utilisons un rayon de fenêtre de 40, puis 30, puis 20 et enfin 10 pixels, en réalisant 2 itérations à chaque fois. La pyramide multi-échelle utilise 2 niveaux.

Afin de pouvoir vérifier la pertinence de ce choix de paramètres, indépendamment des difficultés liées à l'utilisation de modalités différentes, nous donnons également les résultats de GeFolki en optique/optique, avec le même choix de paramètres et sur les mêmes images [2]. Pour le cas optique/optique, comme les deux images obtenues sont issues de la même acquisition, nous appliquons un bruit additif gaussien sur l'une des images — le niveau de bruit est choisi de sorte à avoir un rapport signal à bruit de 5%.

C.5.2 Résultats

Le tableau C.1 présente l'erreur moyenne obtenue avec les différentes méthodes. Le résultat de PWC-Net-multimodal sur le cas multimodal est bien meilleur (EPE divisée par 2) que celui de GeFOLKI pour les deux bases de test, même s'il est loin de la précision atteignable par GeFolki sur des images issues du même capteur (cas optique/optique). Les figures C.6 et C.7 montrent quelques résultats qualitatifs. On peut observer les différences importantes entre les images des deux modalités (inversions de contrastes) notamment dans le second exemple, où les régions urbaines sont dominées par des échos forts dans l'image SAR. Sur le cas optique/optique, GeFolki donne une estimation de bonne qualité, en revanche les champs obtenus n'ont que peu de rapport avec la VT sur les données multimodales. Dans ce cas les champs de déplacements estimés par PWC-Net-multimodal approximent bien le champ VT, malgré une perte de résolution spatiale importante. Cette évaluation est cependant effectuée sur des données très proches de celles utilisées pour l'entraînement : bien que nous ayons mis de côté des images pour la validation, celles-ci sont issues des mêmes modalités d'acquisition que les données d'entraînement, et les champs de déplacements sont générés selon le même procédé. L'évaluation est donc un peu biaisée, puisqu'il s'agit d'un contexte sur lequel PWC-Net-multimodal s'est spécialisé. Les résultats restent très encourageants et incitent à poursuivre les évaluations notamment vers des scènes ou des données différentes, et plus généralement, à approfondir la question de l'utilisation de telles méthodes basées sur l'apprentissage profond pour réaliser le recalage d'images hétérogènes de télédétection.

C.6 Conclusion

Nous avons collecté des images optique et SAR superposables, ainsi que le modèle numérique de terrain associé à chaque paire d'images, et proposé un protocole d'entraî-

2. Ceci est rendu possible par notre processus de génération des données : à partir du même couple optique/SAR superposable et d'un champ de déplacement on peut générer aussi bien une paire à recaler optique/optique qu'une paire à recaler optique/SAR, et ces deux paires présentent le même champ de déplacement.

nement supervisé basé sur

- une génération de champs de déplacements à partir du relief.
- le rééchantillonnage des images optique et SAR afin d'obtenir des paires optique/optique, SAR/SAR et optique/SAR de la même zone et présentant le même champ de déplacement.
- l'utilisation d'une fonction de coût à trois termes, pénalisant simultanément trois tâches de recalage (optique, SAR et hétérogène) réalisées avec le même réseau de neurones.

Nous avons également proposé l'architecture PWC-Net-multimodal, basée sur PWC-Net mais pouvant apprendre des représentations différentes pour les deux voies d'entrée. L'architecture PWC-Net-multimodal entraînée selon le protocole proposé permet le recalage des couples optique/SAR des ensembles de validation associés à notre base d'entraînement, avec une précision nettement supérieure à celle obtenue avec GeFolki.

Ces résultats sont très encourageants. Pour poursuivre ces travaux, il faudra évaluer la méthodes sur d'autres données, et éventuellement réfléchir à une génération plus réaliste des champs de déplacements d'entraînement.

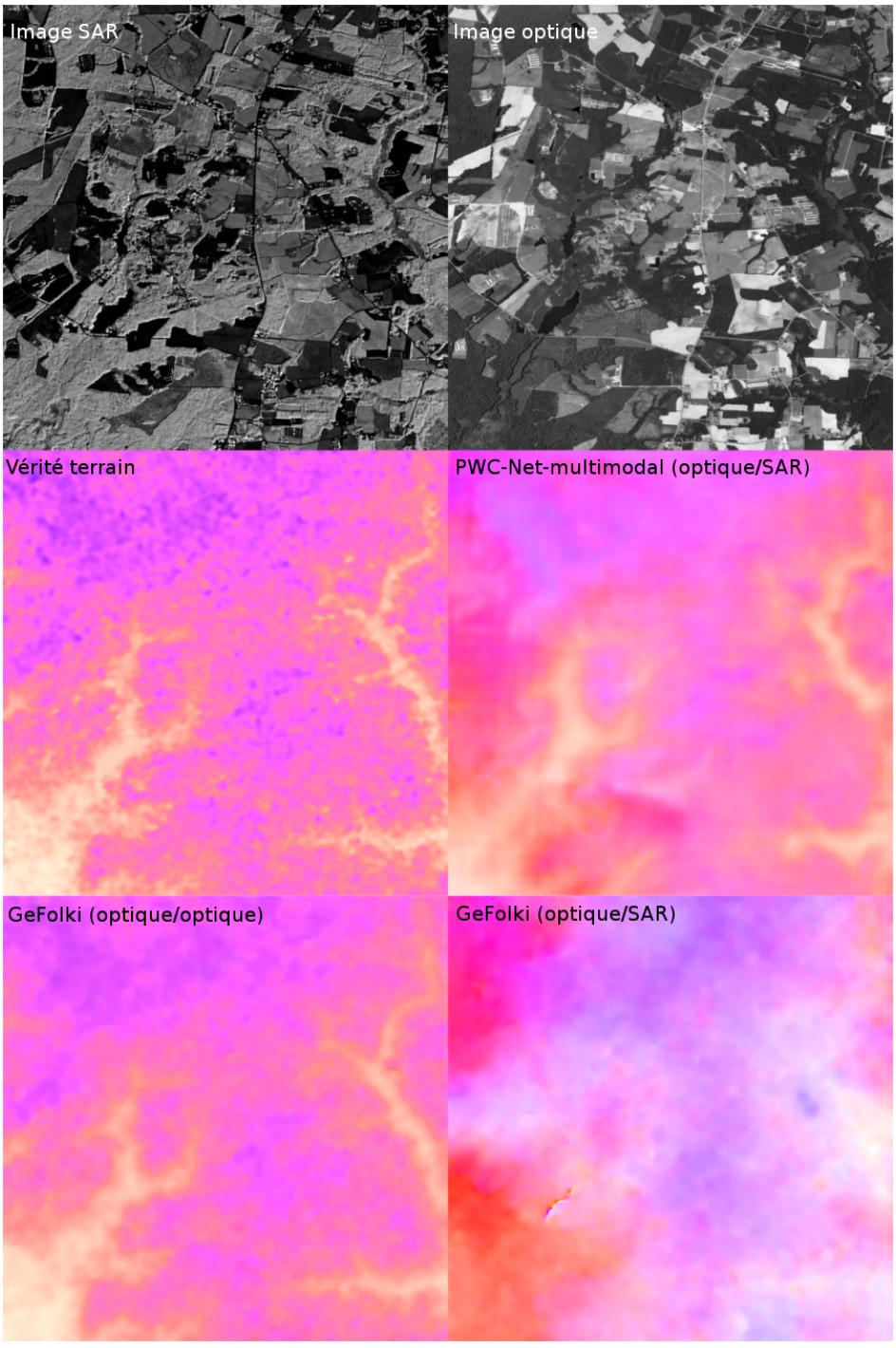

FIGURE C.6 – Résultats qualitatifs issus de l'évaluation sur l'ensemble de validation de la base d'images aériennes.

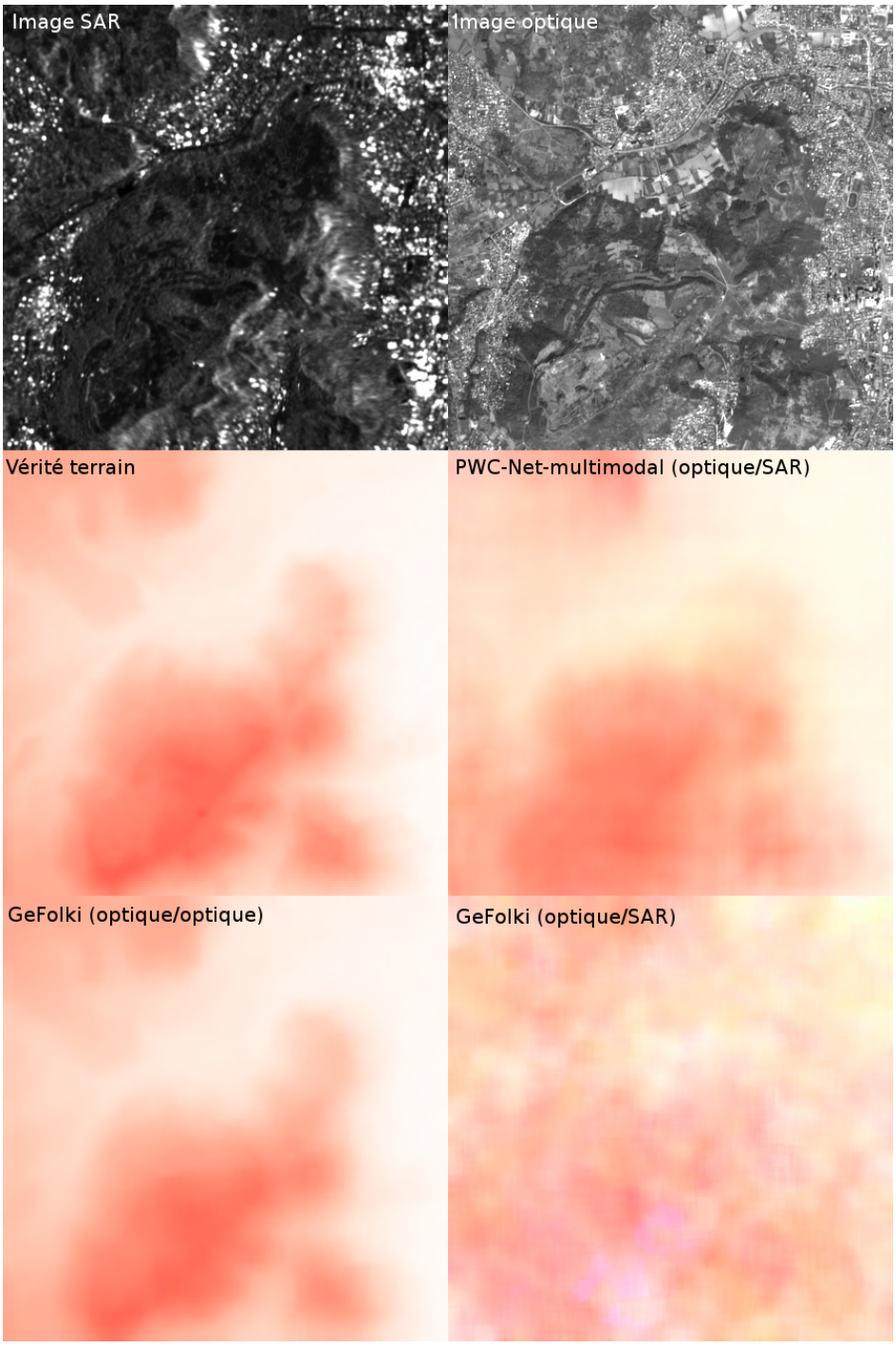

FIGURE C.7 – Résultats qualitatifs issus de l'évaluation sur l'ensemble de validation de la base d'images satellitaires.

C.7 Références

BRIGOT, G., E. COLIN-KOENIGUER, A. PLYER et F. JANEZ. 2016, «Adaptation and evaluation of an optical flow method applied to coregistration of forest remote sensing images», *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, nº 7, p. 2923–2939. XVI

SUN, D., X. YANG, M.-Y. LIU et J. KAUTZ. 2018, «PWC-Net : CNNs for optical flow using pyramid, warping, and cost volume», dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 8934–8943. XX

ZABIH, R. et J. WOODFILL. 1994, «Non-parametric local transforms for computing visual correspondence», dans *European conference on computer vision*, Springer, p. 151–158. XVI

Annexe D

Liste des acronymes

CNN réseau de neurones convolutif ou *Convolutional Neural Network*. 28, 29, 39, 85, 98, 106, 108

EPE erreur *endpoint*. II, V, XXI, 19, 32, 87, 101, 104, 107, 114

GPU processeur graphique ou *graphics processing unit*. VI, 27, 113

GRU *Gated Recurrent Unit*. 117–119

LKFT *Lucas-Kanade Fluid Trajectories*. 7, 9, 41, 44–54, 59, 65, 141, 142, 148

LSTM *Long Short-Term Memory*. 117

PCA analyse en composantes principales ou *Principal Component Analysis*. 44

PIV vélocimétrie par images de particules ou *Particle Image Velocimetry*. 3, 4, 27, 41, 44, 46, 48, 64, 65, 138, 140, 148

UV ultraviolet. 18

VT vérité terrain. II–IV, XV, XX, XXI, 18, 19, 32, 46, 48–50, 59, 61–64, 68, 70, 71, 74, 75, 77, 80, 81, 83–87, 92, 97, 109–111, 137

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Approches par apprentissage pour l'estimation de mouvement multiframe en vidéo

Mots clés : Flot optique, Vision par ordinateur, Apprentissage profond, Apprentissage par ordinateur

Résumé : Ce travail porte sur l'exploitation de l'information temporelle sur une séquence de plus de deux images pour l'estimation du flot optique, défini comme le champ dense (en tout pixel) des mouvements apparents dans le repère image. Nous étudions d'une part l'utilisation d'une base de modèles temporels, appris par analyse en composantes principales à partir des données étudiées, pour modéliser la dépendance temporelle du mouvement. Cette première étude se focalise sur le contexte de la vélocimétrie par images de particules en mécanique des fluides. D'autre part, le nouvel état de l'art de l'estimation de flot optique ayant récemment été établi par des méthodes basées sur l'apprentissage profond, nous entraînons des réseaux de neurones convolutifs à estimer le flot optique en profitant de la continuité temporelle, dans le cas de séquences d'images naturelles. Nous proposons ensuite STaRFlow, un réseau de neurones convolutif exploitant une mémoire de l'information du passé au moyen d'une récurrence temporelle. Par application répétée d'une même cellule récurrente, les mêmes paramètres appris sont utilisés pour les différents instants considérés et pour les différents niveaux d'un processus multi-échelle. Cette architecture est plus légère que les réseaux concurrents tout en conférant à STaRFlow des performances à l'état de l'art. Au fil de nos travaux, nous mettons en évidence plusieurs cas où l'utilisation de l'information temporelle permet d'améliorer la qualité de l'estimation, en particulier en présence d'occultations, lorsque la qualité image est dégradée (flou, bruit), ou encore dans le cas d'objets fins.

Title : Learning approaches for multi-frame motion estimation

Keywords : Optical Flow, Computer Vision, Deep learning, Machine learning

Abstract : This work concerns the use of temporal information on a sequence of more than two images for optical flow estimation. Optical flow is defined as the dense field (in any pixel) of the apparent movements in the image plane. We study on the one hand the use of a basis of temporal models, learned by principal component analysis from the studied data, to model the temporal dependence of the movement. This first study focuses on the context of particle image velocimetry in fluid mechanics. On the other hand, the new state of the art of optical flow estimation having recently been established by methods based on deep learning, we train convolutional neural networks to estimate optical flow by taking advantage of temporal continuity, in the case of natural image sequences. We then propose STaRFlow, a convolutional neural network exploiting a memory of information from the past by using a temporal recurrence. By repeated application of the same recurrent cell, the same learned parameters are used for the different time steps and for the different levels of a multiscale process. This architecture is lighter than competing networks while giving STaRFlow state-of-the-art performance. In the course of our work, we highlight several cases where the use of temporal information improves the quality of the estimation, in particular in the presence of occlusions, when the image quality is degraded (blur, noise), or in the case of thin objects.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France